Autorskie rozwiązanie klastrowego systemu bazodanowego MySQL w środowisku OpenSSI

RSO - Rozproszone Systemy Informacyjne

Zespół RSO1 w składzie:
Krzysztof Fabjański (kfabjans@elka.pw.edu.pl)
Janusz Nowacki (J.Nowacki@elka.pw.edu.pl)
Rafał Paluch (rafal.paluch@gmail.com)
Sebastian Strzelak (sebu@multipleks.pl)
Adam Śniegórski (sniegorscy@chello.pl)
Janusz Twardziak (J.Twardziak@elka.pw.edu.pl)

15. czerwca 2006

Spis treści

1 Wprowadzenie

Niniejsza dokumentacja powstała w ramach realizacji zadania projektowego, którego celem było rozpoznanie systemu bazodanowego MySQL, środowiska klastrowego OpenSSI. W dalszej fazie realizacji zadaniem zespołu było opracowanie koncepcji, implementacja oraz przeprowadzenie testów klastrowego systemu bazodanowego opartego o MySQL we wspomnianym środowisku OpenSSI. Projekt podzielony został na trzy fazy.

I faza projektu dotyczyła teoretycznego opracowania zagadnień związanych z projektem, w tym:

1. systemu bazodanowego MySQL,
2. środowiska OpenSSI,
3. metod zapewniania niezawodności w systemach rozproszonych.

Efektem realizacji tej fazy projektu była również część praktyczna - przygotowanie paczki instalującej standardowe rozwiązanie MySQL w katalogu użytkownika.

Wymogiem II fazy projektu była realizacja następujących zadań:

1. przygotowanie paczki instalacyjnej rozwiązania MySQL z rozszerzeniami klastrowymi - MySQL Cluster,
2. przygotowanie zestawu testów wydajnościowych i naciskowych systemu bazodanowego opartego o rozwiązania MySQL,
3. opracowanie autorskiej koncepcji rozwiązania klastrowego rozszerzonego o kod wykorzystujący optymalnie środowisko OpenSSI.

Wymogiem III fazy projektu była realizacja zadań:

1. implementacji zaprojektowanego rozwiązania autorskiego,
2. przeprowadzenie testów wydajnościowych i naciskowych oraz na przygotowanych instalacjach MySQL,
3. opracowanie autorskiej koncepcji rozwiązania klastrowego rozszerzonego o kod wykorzystujący optymalnie środowisko OpenSSI.

Końcowym efektem prac zespołu są paczki instalacyjne:

- standardowego serwera MySQL bez rozszerzeń klastrowych,
- dostarczanego przez MySQL AB rozwiązania z rozszerzeniami klastrowymi - MySQL Cluster,
- autorskiego rozwiązania opartego o MySQL z rozszerzeniami klastrowymi uwzględniającego możliwości systemu OpenSSI.

Odrębną ścieżką prac projektowych tej było opracowanie i przeprowadzeni zestawu testów, zarówno wydajnościowych jak i naciskowych, pozwalających wyrokować o cechach poszczególnych rozwiązań MySQL używanych w trakcie pracy nad projektem. Stworzone narzędzie testujące pozwoliło na weryfikację realizacji celów postawionych przed zespołem projektowym.

Niniejszy dokument stanowi sprawozdanie z wykonanych prac projektowych. Sekcje 2-3 odnoszą się do opracowań teoretycznych kluczowych zagadnień związanych z projektem. Sekcje 4-6 odnoszą się do szczegółów instalacji:

- standardowego serwera MySQL bez rozszerzeń klastrowych,
- rozwiązania MySQL Cluster z rozszerzeniami klastrowymi,
- autorskiego rozwiązania zaimplementowanego w ramach projektu.

Sekcja siódma omawia szczegółowo koncepcję rozwiązania autorskiego uwzlędniającego specyfikę środowiska OpenSSI. Sekcja ósma omawia zaprojektowany zestaw testów wydajnościowych i naciskowych. W sekcji dziewiątej zaś przedstawiono wyniki testów wydajnościowych przeprowadzonych na instalacji standardowej MySQL.

Dokumentowi temu towarzyszą wszystkie stworzone w ramach projektu dokumenty i opracowania.

W trakcie realizacji projektu wykorzystano narzędzie kontroli wersji Subsersion. Przygotowanie opracowania dotyczącego tego systemu oraz zarządzanie repozytorium projektowym wchodziło w zakres projektu.

2 Realizacja klastrowości w istniejących rozwiązaniach MySQL

Celem niniejszego opracowania jest opis rozwiązań istniejących w ramach MySQL umożliwiających realizację klastrowości, zwiększenia niezawodności i wydajności serwera bazy danych oraz analiza pod kątem możliwości zastosowania w projektowanym rozwiązaniu przeznaczonym na klaster OpenSSI.

MySQL oferuje rozwiązanie klastrowe MySQL Cluster, które zostanie omówione w pierwszej części opracowania. Jest to rozwiązanie znajdujące się w fazie rozwoju i wiele jego cech jak na przykład wymagana ilość pamięci operacyjnej w klastrze sugeruje, iż jest pole do działania w zakresie stworzenia rozwiązania podobnego, lecz być może bardziej efektywnego pod kątem wykorzystania zasobów klastra przy wykorzystaniu wbudowanych w MySQL mechanizmów.

W dalszej części opracowania szczegółowo omówione zostanie zagadnienie replikacji danych w MySQL, które jest podstawowym mechanizmem zapewnienia wspomnianych pożądanych cech.

2.1 Istniejące rozwiązanie klastrowe - MySQL Cluster

MySQL Cluster to wersja MySQL, charakteryzująca się wysoką dostępnością i wysoką nadmiarowością danych, dostosowana do specyfiki rozproszonych środowisk obliczeniowych. Wykorzystuje mechanizm składowania danych `NDB Cluster` umożliwiający uruchomienie kilku serwerów MySQL na klastrze. Wersja MySQL Cluster dostępna jest obecnie na platformy Linux, Mac OS X oraz Solaris.

Szczegółowe informacje informacje o wersji MySQL Cluster można znaleźć w dokumentacji MySQL - **Rozdział 16. MySQL Cluster**.
`http://dev.mysql.com/doc/refman/5.1/en/ndbcluster.html`

2.2 Omówienie rozwiązania klastrowego MySQL Cluster

MySQL Cluster według dokumentacji jest technologią umożliwiającą realizację założeń klastrowej bazy danych utrzymywanej w pamięci operacyjnej. Zaletą tego rozwiązania jest możliwość pracy na niedrogim sprzęcie i brak specyficznych wymagań co do oprogramowania i sprzętu. Eliminuje również istnienie pojedynczych punktów awarii, jako że z założenia każdy komponent ma własną pamięć oraz przestrzeń dyskową.

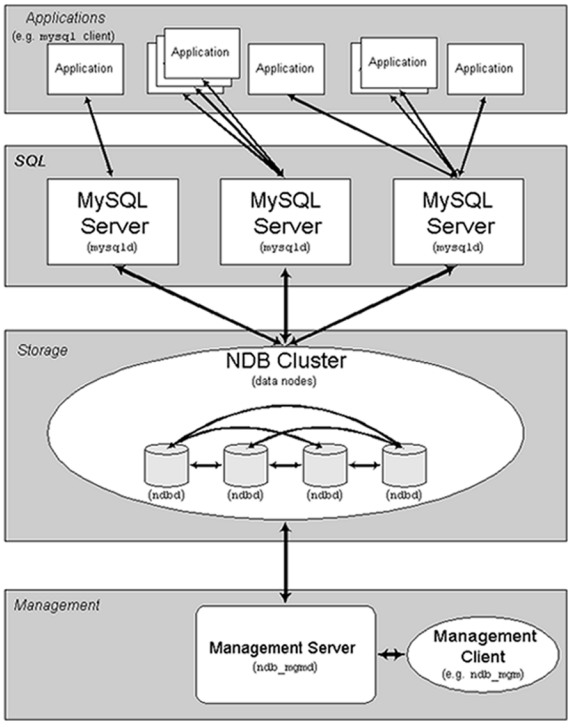
MySQL Cluster łączy standardowy serwer MySQL z klastrowy mechanizm składowania danych NBD, utrzymywany w całości w pamięci operacyjnej. W tym sensie MySQL Cluster to połączenie MySQL i mechanizmu składowania NBD.

Klaster MySQL składa się z grupy komputerów, z których na każdym uruchomiona jest pewna liczba procesów takich jak:

- serwery MySQL,

- węzły danych NDB *(ang. data node)*,
- węzły zarządzające *(ang. management node)*,
- ewentualne specjalistyczne oprogramowanie dostępu do danych.

Rysunek 1 obrazuje relację między tymi komponentami.



Rysunek 1: Komponenty rozwiązania klastrowego MySQL Cluster

Wymienione procesy współpracują, tworząc rozwiązanie MySQL Cluster. Tabele danych w mechaniźmie NDB przechowywane są na węzłach danych i są dostępne dla wszystkich serwerów MySQL w klastrze. Jeśli jedna aplikacja zmieni porcję danych w klastrze, wszystkie serwery odpytujące te dane widzą zmiany natychmiast.

Dane przechowywane w węzłach danych mogą być replikowane, a więc rozwiązanie klastrowe jest odporne na awarię pojedynczego węzła. Jedyną konsekwencją takiej awarii jest cofnięcie części aktualnie wykonywanych transakcji.

2.2.1 Zarys rozwiązań zastosowanych w MySQL Cluster

Mechanizm składowania danych NDB Cluster NDB jest mechanizmem składowania danych oferującym wysoką dostępność i trwałość danych. Dane w NDB przechowywane są w pamięci, z możliwością utrzymywania kopii zapasowych na dyskach twardych. Mechanizm ten może zostać skonfigurowany opcjami zapewniającymi wysoką odporność na awarię i równoważenie obciążenia.

Wpęzły w MySQL Cluster Mechanizmy realizujące klastrowość są w obecnej wersji konfigurowane niezależnie od serwerów MySQL. Każda część klastra nazywana jest węzłem. W MySQL Cluster jako węzęł rozumie się nie pojedyncza maszynę, lecz proces. Na jednym komputerze z klastra może istnieć wiele węzłów. Komputer w klastrze nazywany jest hostem węzłów.

W minimalnej konfiguracji MySQL Cluster istnieją co najmniej trzy węzły:

- **wezeł zarządzający**: rolą węzła tego typu jest zarządzanie pozostałymi węzłami w ramach MySQL Cluster, tj. dostarczanie danych konfiguracyjnych, uruchamianie i zatrzymywanie węzłów, tworzenie kopii zapasowych, itd.
- **węzeł danych**: węzeł tego typu przechowuje dane dostępne w całym klastrze. Istnieje tyle węzłów danych ile replik. Przykładowo, dla dwóch replik, z których każda składa się z dwóch fragmentów (partycji) potrzeba czterech węzłów danych. Nie ma konieczności tworzenia więcej niż jednej repliki.
- **węzeł SQL**: węzeł tego typu zapewnia dostęp do danych klastra. W MySQL Cluster jest to standardowy serwer MySQL ze skonfigurowanym mechanizmem składowania danych NDB. Węzeł SQL jest więc procesem serwera MySQL uruchumionym z opcją `--ndbcluster`.

Konfiguracja klastra polega na konfiguracji każdego pojedynczego węzła w klastrze i zestawieniu połączeń pomiędzy węzłami. MySQL Cluster został zaprojektowany przy założeniu że wszystkie węzły przechowujące dane mają jednakową moc obliczeniową, dostępną pamięć i przepustowość łącza. W celu ułatwienia konfiguracji wszystkie dane konfiguracyjne umieszczone są w pojedynczym pliku konfiguracyjnym.

Węzeł zarządzający utrzymuje plik konfiguracyjny oraz plik dziennika dla całego klastra. Każdy węzeł pobiera dane konfiguracyjne z węzła zarządzającego, wymaga więc informacji, gdzie znajduje się węzeł zarządzający.

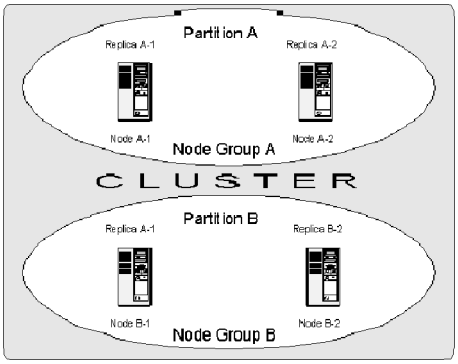
Pojęcia węzłów, grup węzłów, replik i partycji Ta część opracowania pokazuje w jaki sposób MySQL Cluster dzieli i duplikuje dane. Poniżej przedstawiono kluczowe pojęcia pomagające zrozumieć działanie klastra:

- **Węzeł danych**: Proces, przechowujący *replikę*, tj. kopię *partycji* przypisaną do grupy węzłów, której jest częścią.

  Każdy węzeł danych jest z reguły umieszczony na innej maszynie. Jednak jest możliwe umieszczenie więcej niż jednego węzła danych na jednym komputerze, posiadającym wiele procesorów. W takim wypadku możliwe jest uruchomienie jednego węzła danych na jednym procesorze.

- **Grupa węzłów**: Grupa węzłów (danych) składa się z jednego lub więcej węzła danych i przechowuje fragment (partycję) lub zbiór replik. Obecnie każda grupa węzłów musi składać się z tej samej liczby węzłów.

- **Partycja**: Fragment danych przechowywanych przez klaster. Liczba partycji w klastrze odpowiada liczbie grup w klastrze, każda grupa węzłów odpowiedzialna jest za utrzymanie przynajmniej jednej kopii przypisanej partycji, dostępnej dla całego klastra.
- **Replika**: Kopia partycji klastra. Każdy węzeł danych w grupie węzłów przechowuje jedną replikę.



Rysunek 2: Komponenty rozwiązania klastrowego MySQL Cluster

Rysunek 2 ilustruje klaster MySQL Cluster z czterema węzłami danych, uporządkowanymi w dwie grupy, po dwa węzły w każdej z grup. Dane przechowywane w klastrze podzielone są na dwie partycje. Każda partycja przechowywana jest - w dwóch kopiach - w jednej grupie węzłów. Dane tworzące partycję A, przechowywane na węźle A-1 są identyczne jak dane na węźle A-2. Analogicznie, dane tworzące partycję B, przechowywane są na węzłach B-1 oraz B-2.

Jak widać założenie wysokiej dostępności jest realizowane w następujący sposób. Jak długo każda z grup tworzących klaster ma co najmniej jeden działający węzeł, tak długo cały klaster posiada kompletny zestaw danych i może funkcjonować prawidłowo. Każda kombinacja co najmniej jednego węzła przechowującego replikę partycji A z co najmniej jednym węzłem przechowującym replikę partycji B tworzy zupełny obraz danych przechowywanych przez klaster. Jednak awaria obu węzłów z jednej grupy powoduje, że klaster nie posiada kompletu danych i nie może dalej pełnić swojej roli.

2.2.2 Wybrane zagadnienia opisane w MySQL Cluster FAQ

Różnice między rozwiązaniem MySQL Cluster a użyciem replikacji W przypadku systemu opartego na replikacji serwer nadrzędny *(ang. master)* uaktualnia jeden lub więcej serwer podrzędny *(ang. slave)*. Transakcje w MySQL wykonywane są sekwencyjnie, powolne wykonanie transakcji może spowodować opóźnienie serwera podrzędnego w stosunku do serwera nadrzędnego. Tak więc istnieje ryzyko, iż w przypadku awarii serwera nadrzędnego, serwery podrzędne nie będą miały informacji o kilku ostatnich transakcjach. W bezpiecznym z punktu widzenia transakcji mechaniźmie składowania `InnoDB` transakcja będzie wykonana w całości na serwerze podrzędnym lub zostanie wycofana. Jednak replikacja nie gwarantuje spójności danych na serwerze nadrzędnym i serwerach podrzędnych w każdym punkcie czasu.

W rozwiązaniu MySQL Cluster wszystkie węzły danych są zawsze zsynchronizowane. Transakcje zatwierdzone na jednym węźle danych są zatwierdzane na wszystkich pozostałych węzłach. W przypadku awarii jednego z węzłów, pozostałe węzły pozostają w spójnym stanie.

Podsumowując, replikacja w MySQL Cluster jest synchroniczna podczas gdy w standardowym MySQL asynchroniczna.

Zapotrzebowanie na pamięć RAM w MySQL Cluster Obecnie MySQL Cluster jest systemem przechowującym bazę danych w pamięci operacyjnej. Oznacza to, że wszystkie tabele przechowywane są w pamięci RAM. Z tego względu, jeśli dane mają łączny rozmiar 1GB i stosuje się jedną replikę w klastrze, potrzeba 2GB pamięci RAM (nie uwzgledniając pamięci RAM dla systemu operacyjnego i działających usług).

Jeśli węzeł danych wyczerpie dostępną pamięć RAM, system podejmie próbę wykorzystania pamięci swap. Jednak w najlepszym wypadku spowoduje to znaczny spadek wydajności, w najgorszym doprowadzi do wyłączenia węzła danych z powodu długiego czasu odpowiedzi. Poleganie na pamięci swap w środowisku produkcyjnym nie jest zalecane.

Ilość pamięci wymagana dla każdego z węzłów danych może być oszacowana według następującego wzoru:

$$(SizeofDataBase * NumberOfReplicas * 1.1)/NumberOfDataNodes$$

Warto również wspomnieć, że wszystkie węzły danych powinny mieć taki sam rozmiar dostępnej pamięci RAM. Żaden węzeł danych nie może wykorzystywać więcej pamięci niż wynosi najmniejszy rozmiar pamięci dostępnej dla węzła danych w klastrze.

2.2.3 Więcej informacji o MySQL Cluster

Przykład ilustrujący zestawienie klastra MySQL Cluster w dokumentacji MySQL - **Sekcja 16.3** *Simple Multi-Computer How-To*
`http://dev.mysql.com/doc/refman/5.1/en/multi-computer.html?ff=nopfpls`

Opis konfiguracji rozwiązania MySQL Cluster w dokumentacji MySQL - **Sekcja 16.4** *MySQL Cluster Configuration*
`http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-configuration.html`

Opis procesów składających się na MySQL Cluster w dokumentacji MySQL - **Sekcja 16.5** *Process Management in MySQL Cluster*
`http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-process-management.html`

Opis zarządzania klastrem w dokumentacji MySQL - **Sekcja 16.6** *Management of MySQL Cluster*
`http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-management.html`

Opis mechanizmu przechowywania danych MySQL Cluster na dysku w dokumentacji MySQL - **Sekcja 16.8** *MySQL Cluster Disk Data Storage*
`http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-disk-data.html`

Lista znanych ograniczeń rozwiązania MySQL Cluster w dokumentacji MySQL - **Sekcja 16.10** *Known Limitations of MySQL Cluster*
`http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-limitations.html`

2.3 Mechanizmy replikacji w MySQL

Termin *replikacja*, rozumiany jako utrzymywanie wielu kopii (replik) tych samych danych na wielu systemach plików, odnosi się do zapewnienia niezawodności (rozproszonego) systemu informatycznego oraz zwiększenia jego odporności na awarię przy pomocy rozwiązań sprzętowych lub poprzez zaimplementowanie tej funkcjonalności w działającym na danym systemie oprogramowaniu. Przykładami sprzętowego rozwiązania replikacji są:

- macierz dyskowa typu RAID (począwszy od RAID-1),
- rozproszone systemy plików: np. AFS, NFSv4, CODA (`http://coda.cs.cmu.edu/`).

Poprzez zastosowanie replikacji w rozproszonym systemie komputerowym uzyskuje się:

- **zwiększenie dostępności** danych poprzez umożliwienie dostępu do danych w przypadku niedostępności pojedynczych replik,
- **zwiększenie efektywności** poprzez zmniejszenie opóźnień w dostępie do danych (lokalne repliki) i zwiększenie przepustowości (wiele serwerów pracujących współbieżnie).

Replikacja w systemach bazodanowych odbywa się zwykle na bazie relacji master-slave między oryginałem a replikami. Serwer nadrzędny *(ang. master)* tworzy dziennik (log) zmian, który następnie rozpowszechniany jest do serwerów podrzędnych *(ang. slave)*. Serwery podrzędne powiadamiają o otrzymaniu informacji o zmianach, zezwalając tym samym na wysyłanie kolejnych zmian.

Niniejsze opracowanie omawia mechanizm replikacji zaimplementowany w systemie RDBMS MySQL.

2.3.1 Omówienie mechanizmu replikacji asynchronicznej w MySQL

RDBMS MySQL umożliwia jednokierunkowoą, asynchroniczną replikację danych, w której jeden serwer pełni rolę serwera nadrzędnego *(ang. master)*, podczas gdy inny serwer lub grupa serwerów są serwerami podrzędnymi *(ang. slave)*. Jest to zasadnicza różnica w odniesieniu do *synchronicznej* replikacji zastosowanej w rozwiązaniu MySQL Cluster.

W przypadku replikacji z jednym serwerem nadrzędnym *(ang. single-master replication)*, serwer nadrzędny zapisuje wszystkie zmiany w plikach logów binarnych *(ang. binary log files)* i trzyma

indeks tych plików w celu śledzenia obiegu logów. Serwer podrzędny, łącząc się z serwerem nadrzędnym, informuje go o pozycji ostatniego zakończonego sukcesem uaktualnienia zmian. Na tej podstawie otrzymuje od serwera nadrzędnego informacje o wszystkich zmianach, jakie miały miejsce od tego czasu. Po ich zastosowaniu do własnej kopii danych przechodzi w stan oczekiwania na nowe zmiany.

Serwer podrzędny może służyć jednocześnie za serwer nadrzędny. Możliwe jest więc tworzenie łancuchów w postaci:

```
A -> B -> C -> A
```

Istnieje możliwość zastosowania replikacji z wieloma serwerami nadrzędnymi *(ang. mupltiple-master replication)*. W tym wypadku należy jednak zwrócić uwagę na dodatkowe kwestie związane z możliwymi kolizjami kolumn typu `AUTO_INCREMENT`. Serwery nadrzędne mogą próbować użyć tych samych wartości `AUTO_INCREMENT` podczas dodawania rekordów.

Można się przed tym zabezpieczyć ustawiając odpowiednio zmienne systemowe `auto_increment_increment` oraz `auto_increment_offset`. Obie zmienne mają minimalną (a zarazem domyślną) wartość 1 a ich wartość maksymalna to 65,535.

Ustawienie wspomnianych zmiennych wpływa na kolumny typu `AUTO_INCREMENT` w następujący sposób:

- `auto_increment_increment` kontroluje krok zwiększania kolejnych wartości `AUTO_INCREMENT`.
- `auto_increment_offset` ustala wartość początkową dla kolumn typu `AUTO_INCREMENT`.

Istnieje więc możliwość ustawienia na każdym z serwerów nadrzędnych wartości nie powodujących konfliktów w numeracji rzędów. Zakładając `N` serwerów nadrzędnych ustawienie zmiennych na każdym z nich powinno wyglądać następująco:

- `auto_increment_increment` powinien wynosić `N` na każdym serwerze nadrzędnym,
- `auto_increment_offset` powinien mieć inną wartość z zakresu 1,2,...,`N` na każdym z serwerów nadrzędnych.

Korzyści płynące z zastosowania replikacji w MySQL W efekcie zastosowania mechanizmu replikacji w MySQL uzyskuje się korzyści dotyczące odporności na awarię, szybkości działania oraz zakresu administracji systemem:

- Odporność na awarię zwiększana jest poprzez zestawienie master-slave. W przypadku problemów z serwerem nadrzędnym istnieje możliwość przełączenia się na kopię zapasową dostępną na serwerze podrzędnym.

- Krótszy czas odpowiedzi dla aplikacji klienckich mozna uzyskać poprzez rozłożenie obciążenia przetwarzaniem zapytań pomiędzy serwer nadrzędny a serwery podrzędne. Zapytania typu `SELECT` mogą być przekazywane serwerom podrzędnym w celu zmniejszenia obciążenia przetwarzaniem zapytań na serwerze nadrzędnym.

  UWAGA! Zapytania modyfikujące dane powinny być zawsze wykonywane na serwerze nadrzędnym. Przyczyny zostaną omówione poniżej.

- Korzyści administracyjne polegają na tworzeniu kopii zapasowych w czasie rzeczywistym, bez ingerencji w działanie serwera nadrzędnego. Przy zastosowaniu replikacji serwer nadrzędny kontynuuje normalną pracę a jednocześnie tworzone są kopie zapasowe na serwerach podrzędnych.

Replikacja w MySQL a zapytania modyfikujące dane Stosując replikację w MySQL należy zadbać o to by wszystkie zmiany w replikowanych tabelach były wykonywane na serwerze nadrzędnym. W przeciwnym wypadku należy mieć na uwadze możliwość wystąpienia konfliktów między zmianami dokonanymi przez użytkowników na tabelach przechowywanych na serwerze nadrzędnym a zmianami na tabelach przechowywanych na serwerach podrzędnych.

W MySQL istnieją dwa typy replikacji: replikacja oparta na zapytaniach i replikacja oparta na rzędach. Szczegółowe różnice zostanę omówione w dalszej części opracowania, na tym etapie wystarczy wiedzieć, że w przypadku replikacji opartej na zapytaniach w binarnych logach serwera nadrzędnego zapisywane są zapytania modyfikujące dane, natomiast w przypadku replikacji opartej na rzędach serwer nadrzędny zapisuje w logu binarnym zmiany jakie dokonywane są na poszczególnych rzędach.

Wracając do zagadnienia zapytań modyfikujących dane na wykonywanych na serwerach podrzędnych, weźmy pod uwagę następujący scenariusz, w którym na serwerze podrzędnym do jednej z tabel dodawany jest nowy rząd, podczas gdy zaraz później na serwerze nadrzędnym ta sama tabela jest opróżniana:

```
slave> INSERT INTO tbl VALUES (1);
master> DELETE FROM tbl;
```

Serwer nadrzędny nie ma informacji o operacji `INSERT` wykonanej na serwerze podrzędnym. W przypadku replikacji opartej na zapytaniach, tablica `tbl` zostanie opróżniona zarówno na serwerze nadrzędnym jak i na podrzędnym zaraz po synchronizacji. Jednak w przypadku zastosowania replikacji opartej na rzędach, po synchronizacji na serwerze podrzędnym usunięte zostaną tylko rzędy, o których informacje posiadał serwer nadrzędny. W efekcie w systemie pojawia się niespójność danych.

2.3.2 Warianty replikacji w MySQL

Pierwotnie mechanizm replikacji w MySQL opierał się na propagacji zapytań SQL z serwera nadrzędnego do serwerów podrzędnych. Ten rodzaj replikacji określa się mianem replikacji opartej na zapytaniach *(ang. statement-based replication)*. Począwszy od wersji MySQL 5.1.5

udostępniono nową formę replikacji - replikację opartą na rzędach *(ang. row-based replication)*. Zamiast kolejnych zapytań SQL serwer nadrzędny zapisuje w plikach logów binarnych jak zmieniają się poszczególne wiersze tabeli. Począwszy od wersji MySQL 5.1.8 dostępna jest trzecia forma replikacji: replikacja mieszana *(ang. mixed replication)*. W tym przypadku domyślnie stosowana jest replikacja oparta na zapytaniach z przełączaniem na replikację opartą na rzędach w kiedy replikacja oparta na zapytaniach zawodzi. Takie przypadki uwzględniają na przykład replikację procedur składowanych i wyzwalaczy. Pozostałe zostaną omówione w sekcji poświęconej porównaniu obu metod replikacji.

Praca z różnymi wariantami replikacji Podczas budowania paczki MySQL ze źródeł replikacja oparta na rzędach jest domyślnie dostępna, chyba że polecenie `configure`zostanie wywołane z opcją
`--without-row-based-replication`.

Aby replikacja w ogóle była możliwa, musi zostać włączone tworzenie logów binarnych. Należy uruchomić serwer z opcją `--log-bin`.

Serwer MySQL domyślnie używa replikacji opartej na zapytaniach, nawet gdy został skonfigurowany by udostępniać replikację opartą na rzędach.

Warianty replikacji mogą być zmieniane na 3 sposoby: podczas uruchamiania serwera, w trakcie działania serwera, podczas sesji konkretnego klienta.

Podczas uruchamiania serwera wybór wariantu replikacji polega na uruchomieniu serwera z odpowiednią opcją:

- `--binlog-format=row` dla replikacji opartej na rzędach,
- `--binlog-format=mixed` dla replikacji mieszanej,
- `--binlog-format=statement` lub bez podania żadnej opcji dla replikacji opartej na zapytaniach.

Wariant replikacji może być zmieniony w trakcie działania serwera, globalnie dla wszystkich klientów. W tym celu należy zmienić wartość zmiennej globalnej
`binlog_format`. (Aby zmienić wartość zmiennej globalnej należy mieć uprawnienia `SUPER`).

Aby włączyć replikację opartą na zapytaniach należy wydać jedno z dwóch możliwych poleceń:

```
mysql> SET GLOBAL binlog_format = 'STATEMENT';
mysql> SET GLOBAL binlog_format = '1';
```

Aby włączyć replikację opartą na rzędach należy wydać jedno z dwóch możliwych poleceń:

```
mysql> SET GLOBAL binlog_format = 'ROW';
mysql> SET GLOBAL binlog_format = '2';
```

Aby włączyć replikację mieszaną należy wydać jedno z dwóch możliwych poleceń:

```
mysql> SET GLOBAL binlog_format = 'MIXED';
mysql> SET GLOBAL binlog_format = '3';
```

Wariant replikacji nie może być zmieniony (zostanie zgłoszony błąd) w trakcie działania serwera w następujących przypadkach:

- Z procedury składowanej lub wyzwalacza,
- Gdy włączony jest mechanizm `NBD`,
- Gdy w trakcie sesji włączony jest wariant replikacji opartej na rzędach i otwarte są tablice tymczasowe.

UWAGA! W przypadku replikacji opartej na rzędach większość zmian jest zapisywana w binarnym logu w formacie opartym na rzędach. Jednak niektóre zmiany nadal zapisywane są w formacie opartym na zapytaniach. Dotyczy to wszystkich poleceń DDL, np.:
`CREATE TABLE`, `ALTER TABLE`, `DROP TABLE`.

Porównanie wariantów replikacji Zalety replikacji opartej na zapytaniach:

- Sprawdzona technologia dostępna od wersji MySQL 3.23.
- Mniejsze pliki logów binarnych. Gdy zmiany dotyczą wielu rzędów pliki są *dużo* mniejsze.
- Pliki logów binarnych zawierają wszystkie zapytania powodujące zmiany, mogą być więc wykorzystane do audytu bazy danych.
- Pliki logów binarnych mogą być wykorzystane do odtwarzania do punktu w czasie *(ang. Point-In-Time Recovery)*, a nie tylko do celów replikacji.
- Serwery podrzędne mogą być nowszą wersją MySQL z inną strukturą rzędów.

Wady replikacji opartej na zapytaniach:

- Brak możliwości replikacji wszystkich zapytań `UPDATE`: każda niedeterministyczna część zapytania (na przykład użycie funkcji losowych lub niedetrministycznych funkcji użytkownika *«UDF»*) nie może zostać zreplikowana w przypadku replikacji opartej na zapytaniach, podczas gdy replikacja oparta na rzędach replikuje wynik takiej funkcji.
- Zapytania typu `INSERT...SELECT` wymagają większej liczby blokad rzędów niż w przypadku replikacji opartej na rzędach.
- Zapytanie `UPDATE` wymagające przeglądu całej tabeli (brak indeksu na kolumnie wyszukiwania) blokuje znacznie większą liczbę rzędów niż w przypadku replikacji opartej na rzędach.
- Wolniejsza replikacja danych w przypadku złożonych zapytań.

- W przypadku wystąpienia pojedynczego błędu na serwerze podrzędnym, różnica między serwerem nadrzędnym a serwerem podrzędnym będzie rosła z czasem.

Zalety replikacji opartej na rzedach:

- Wszystko może zostać zreplikowane - najbezpieczniejsza forma replikacji. Polecenia DDL replikowane są z wykorzystaniem replikacji opartej na zapytaniach, a zapytania modyfikujące dane replikowane są w formacie opartym na rzędach.
- Technologia wykorzystywana w innych systemach DBMS, wiedza transferowana do MySQL.
- W wielu przypadkach replikacja danych w tabelach posiadających zdefiniowane klucze główne jest szybsza.
- Mniejsza liczba blokad rzędów, stąd większa współbieżność.

Wady replikacji opartej na rzędach:

- Większe (lub *dużo* większe) pliki logów binarnych.
- Pliki logów binarnych zawierają dane, które zostały wycofane w transakcji.
- W przypadku replikacji opartej na rzędach informacja o każdym rzędzie musi być zapisana w logu binarnym. W przypadku replikacji opartej na zapytaniach jedynie zapytanie jest logowane. Jeśli zapytanie zmienia wiele rzędów, replikacja oparta na rzędach powoduje zapisanie znacznie większej liczby danych w pliku logu binarnego. Mogą pojawić się opóźnienia podczas replikacji.
- Brak możliwości analizy logów w celu sprawdzenia wykonywanych zapytań.

2.3.3 Szczegóły implementacyjne replikacji

Replikacja w MySQL zaimplementowana jest przy użyciu 3 wątków (jeden wątek na serwerze nadrzędnym i dwa wątki na serwerze podrzędnym).

Po wydaniu polecenia `START SLAVE` na serwerze podrzędnym tworzony jest wątek I/O, który łączy się z serwerem nadrzędnym i wysyła zapytanie o zmiany zapisane w jego logu binarnym. Serwer nadrzędny tworzy wątek wysyłający zawartość pliku logu binarnego do serwera podrzędnego. Wątek I/O serwera podrzędnego odczytuje zmiany wysłane przez wątek serwera nadrzędnego i kopiuje je do lokalnego pliku logu wybierania *(ang. relay log)*. Trzeci wątek to wątek SQL, tworzony przez serwer podrzędny w celu czytania pliku logu wybierania i wykonywania zmian na lokalnych danych.

Serwer nadrzędny tworzy po jednym wątku dla każdego *podłączonego obecnie* serwera podrzednego. Każdy serwer podrzędny ma własny wątek I/O oraz wątek SQL.

Dzięki użyciu przez serwer podrzędny dwóch wątków, czytanie logu zmian z serwera nadrzędnego i wykonanie zmian na danych lokalnych to dwa niezależne zaadania. W efekcie proces

czytania zmian z serwera nadrzędnego nie jest spowalniany przez wolno wykonujące się zapytania aktualizujące stan bazy. Dla przykładu, jeśli serwer podrzędny przez jakiś czas nie działał, po właczeniu wątek I/O może szybko pobrać zmiany z serwera nadrzędnego nawet jeśli wykonywanie zapytań pozostaje opóźnione. Jeśli serwer podrzędny zakończy działanie przed dokonaniem wszystkich aktualizacji w jego logach wybierania zostaną pobrane już serwera informacje o zmianach gotowe do wykonania przy kolejnym uruchomieniu, bez potrzeby ponownego pobierania ich z serwera nadrzędnego.

UWAGA! Każdy serwer podrzędny śledzi moment ostatniego odczytu logów serwera nadrzędnego. Serwer nadrzędny nie posiada informacji ile ma serwerów podrzędnych ani które z nich mają aktualne dane w określonym punckie czasu.

Wątki serwera nadrzędnego i podrzędnego można zaobserwować wydając polecenie `SHOW PROCESSLIST`. Na serwerze nadrzędnym wątki komunikujące się z serwerami podrzędnymi identyfikowane są jako `Binlog Dump`. Na serwerze podrzędnym wydanie tego samego polecenia spowoduje wyświetlenie informacji o wątku I/O i wątku SQL.

W informacjach zwróconych przez polecenie `SHOW PROCESSLIST` każdy z wątków opisany jest polem tekstowym `State`. Informuje ono o aktualnym stanie wątku - czynnościach jakie wykonał lub aktualnie wykonuje.

Opis stanów wątku serwera nadrzędnego znajduje się w dokumentacji MySQL - **Sekcja 6.4.1** *Replication Master Thread States*
`http://dev.mysql.com/doc/refman/5.1/en/master-thread-states.html`

Opis stanów wątku I/O serwera podrzędnego znajduje się w dokumentacji MySQL - **Sekcja 6.4.2** *Replication Slave I/O Thread States*
`http://dev.mysql.com/doc/refman/5.1/en/slave-io-thread-states.html`

Opis stanów wątku I/O serwera podrzędnego znajduje się w dokumentacji MySQL - **Sekcja 6.4.3** *Replication Slave SQL Thread States*
`http://dev.mysql.com/doc/refman/5.1/en/slave-sql-thread-states.html`

2.3.4 Konfiguracja i uruchamianie replikacji

Przed rozpoczęciem konfiguracji i uruchamiania replikacji należy upewnić się, że logowanie binarne zostało włączone na serwerze nadrzędnym. Aby uruchomić serwer nadrzędny z logowaniem binarnym należy użyć użyć opcji `--log-bin`.

UWAGA! Ważne jest by zrozumieć, że log binarny jest jedynie zapisem zmian rozpoczynającym się w ustalonym punkcie czasu (w momencie włączenia logowania). W związku z tym każdy serwer podrzędny potrzebuje obrazu bazy danych serwera nadrzędnego *w stanie z momentu włączenia logowania binarnego na serwerze nadrzędnym*. W przeciwnym wypadku replikacja nie będzie działać prawidłowo.

Procedura konfiguracji i uruchamiania replikacji Procedura konfiguracji i uruchamiania replikacji na grupie serwerów MySQL została szczegółowo opisana w dokumentacji MySQL 5.1. Nie ma potrzeby powielanie tego opisu. W opracowaniu poniżej znajduje się więc zebranie kroków niezbędnych do zestawienia replikacji. Poszczególne kroki są szczegółowo opisane w dokumentacji MySQL - **Sekcja 6.5** *How to Set Up Replication*.
`http://dev.mysql.com/doc/refman/5.1/en/replication-howto.html`

Dokładny opis poleceń wykorzystywanych w uruchamianiu replikacji dostępny jest w dokumentacji MySQL:

- **Sekcja 13.6.1** *SQL Statements for Controlling Master Servers*
  `http://dev.mysql.com/doc/refman/5.1/en/replication-master-sql.html`
- **Sekcja 13.6.2** *SQL Statements for Controlling Slave Servers*
  `http://dev.mysql.com/doc/refman/5.1/en/replication-slave-sql.html`

Procedura konfiguracji i uruchomienia replikacji

1. Upewnienie się, że wszystkie zainstalowane wersje MySQL są ze sobą kompatybilne.
2. Utworzenie kont na serwerze nadrzędnym umożliwiające serwerom podrzędnym łączenie się w celu pobierania logów binarnych. Kontro musi mieć nadane uprawnienia `REPLICATION SLAVE`.
3. Wykonanie obrazu bazy danych serwera nadrzędnego.
4. Włączenie serwera nadrzędnego z opcją logowania binarnego.
5. Wypełnienie pliku konfiguracyjnego serwerów podrzędnych.
6. Wgranie binarnej kopii bazy danych serwera nadrzędnego na serwerach podrzędnych. W przypadku wykonania kopii zapasowej poleceniem `mysqldump` przejść do kroku następnego.
7. Uruchomienie serwerów podrzędnych.
8. W przypadku wykonania kopii zapasowej serwera nadrzędnego przy użyciu polecenia `mysqldump`, przywrócić kopię zapasową.
9. Ustawienie parametrów serwera nadrzędnego na serwerach podrzędnych.
10. Uruchomienie wątków serwera podrzędnego poleceniem `START SLAVE`.

2.3.5 Zwiększenie wydajności i niezawodności poprzez replikację

Utrata łączności serwera podrzędnego z serwerem nadrzędnym Serwer podrzędny nie musi być podłączony do serwera nadrzędnego cały czas. Po ponownym uruchomieniu serwer podrzędny pobiera wszystkie zmiany zarejestrowane przez serwer nadrzędny od momentu poprzedniej synchronizacji. Konsekwencją takiego stanu rzeczy jest brak gwarancji, że serwer podrzędny jest

zawsze zsynchronizowany z serwerem nadrzędnym. Należy podjąć odpowiednie środki, by wykrywać takie sytuacje.

Istnieje możliwość sprawdzenia jak bardzo opóźniony jest dany serwer podrzędny w stosunku do serwera nadrzędnego. W tym celu należy odczytać wartość kolumny `Seconds_Behind_Master` po wywołaniu polecenia `SHOW SLAVE STATUS` .

Wymuszanie blokowania zapisów na serwerze nadrzędnym do momentu zsynchronizowania danych serwera podrzędnego Zapewnienie synchronizacji danych serwera podrzędnego z serwerem nadrzędnym można zrealizować przy pomocy wymuszenia blokowania zapisów do momentu uaktualnienia wszystkich zmian przez serwer podrzędny. W tym celu należy wykonać następującą procedurę:

1. Na serwerze nadrzędnym należy wydać polecenia:

   ```
   master> FLUSH TABLES WITH READ LOCK;
   master> SHOW MASTER STATUS:
   ```

   Następnie należy zapisać informacje: *nazwa logu* oraz *offset logu*.

2. Na serwerze podrzędnym należy wydać następujące zapytanie z parametrami pobranymi z poprzedniego kroku:

   ```
   slave> SELECT MASTER_POS_WAIT('log_name', log_offset);
   ```

   Zapytanie `SELECT` blokuje serwer nadrzędny do momentu gdy serwer podrzędny otrzyma wybrany plik dziennika i offset.

3. Należy wydać polecenie `UNLOCK TABLES` na serwerze nadrzędnym.

Wykorzystanie replikacji do zwiększenia wydajności systemu W celu zwiększenia wydajności systemu należy uruchomić jeden serwer jako serwer nadrzędny i skierować na niego wszystkie zapisy. Następnie należy skonfigurować jak najwięcej (biorąc pod uwagę ograniczenia przepustowości sieci) serwerów podrzędnych i rozłożyć na nie obciążenie odczytem. Serwery podrzędne mogą być uruchomione z opcjami `–low-priority-updates` , `--skip-inodb` , `--skip-bdb` , `--delay-key-write=ALL` aby uzuskać wzrost wydajności serwerów podrzędnych. Serwery podrzędne w tej sytuacji korzystają z mechanizmu MyISAM, eliminując nakład transakcyjny.

Odciążając serwer nadrzędny z zapytań `SELECT` możemy wielokrotnie zwiększyć liczbę operacji zapisu wykonywanych w jednostce czasu. Należy wówczas jednak zapewnić, że serwery podrzędne będą nadążać z aktualizacją zmian. W przeciwnym wypadku będą one pozostawać opóźnione w stosunku do serwera nadrzędnego lub też będą blokować serwer nadrzędny, co zlikwiduje wzrost wydajności.

2.3.6 Więcej informacji o replikacji w MySQL

Funkcje replikacji i znane problemy zostały opisane w dokumentacji MySQL -
Sekcja 6.8 *Replication Features and Known Problems*
`http://dev.mysql.com/doc/refman/5.1/en/replication-features.html`

Opcje wykorzystywane przy uruchamianiu serwerów podrzędnych opisane zostały w dokumentacji MySQL - **Sekcja 6.9** *Replication Startup Options*
`http://dev.mysql.com/doc/refman/5.1/en/replication-options.html`

Interesujące zagadnienia i kwestie praktyczne poruszone zostały w dokumentacji MySQL - **Sekcja 6.11** *Replication FAQ*
`http://dev.mysql.com/doc/refman/5.1/en/replication-faq.html`

3 System klastrowy OpenSSI

3.1 Czym jest klaster

Klastry budowane są w oparciu o zwykłe, niezależne od siebie komputery, stanowiące węzły systemu klastrowego i nie posiadające współdzielonej pamięci fizycznej. Komputery połączone są szybką siecią komunikacyjną, np. siecią Ethernet. Każdy węzeł klastra posiada niezależny system operacyjny, który za pomocą specjalnie zaprojektowanych rozszerzeń potrafi współpracować z systemami innych węzłów, np. poprzez przezroczyste z punktu widzenia użytkownika rozdzielanie zadań. Użytkownik pracujący przy jednym komputerze klastra ma wrażenie pracy na pojedynczym systemie operacyjnym o niezwykłych możliwościach obliczeniowych. Celem działania klastra jest połączenie komputerów tak, aby zwiększyć moc obliczeniową, niezawodność, czy umożliwić równoległość przetwarzania, np. w odniesieniu do udostępniania zasobów przez działające na komputerach serwery. Zwielokrotnione możliwości przetwarzania danych przez systemy klastrowe przyczyniły się do ich popularności w zastosowaniach wymagających skomplikowanych obliczeń, jak np. renderowanie efektów specjalnych do filmów. Do najpopularniejszych systemów klastrowych należą Beowulf, Mosix, Linux Virtual Server, OpenGFS, Lustre, Oracle Parallel Server, ServiceGuard, Lifekeeper, OpenSSI, OpenMosix, Kerrighed, czy DragonflyBSD.

3.2 Podział systemów klastrowych

Samo słowo ”klaster”tak naprawdę niewiele mówi, ponieważ jest wiele rodzajów systemów klastrowych różniących się sposobem działania i przeznaczeniem. Pewne środowiska informatyczne wyróżniają następujące systemy:

- Klastry wysokiej wydajności (ang. high-performance clusters) - przykładem takiego systemu klastrowego jest Beowulf. Klastry tego typu skonstruowane są tak, aby jak najwydajniej przeprowadzać równoległe obliczenia. Stosowane są m.in. w systemach odkrywania wiedzy czy prognozowania pogody.

- Klastry równoważące obciążenia (ang. load-leveling clusters) - przykładem takiego systemu jest Mosix. Celem ich działania jest równomierne i przezroczyste z punktu widzenia użytkownika dystrybuowanie wykonywanego zadania między pozostałe węzły klastra.

- Klastry udostępniające serwisy WWW (ang. web-service clusters) - przykładem takiego systemu jest Linux Virtual Server. Sposób ich działania jest podobny do działania klastrów równoważących obciążenia. Tym razem jednak, chodzi nie o wykonywanie obliczeń, ale o równomierne przekazywanie zapytań do grupy współpracujących serwerów udostępniających serwisy WWW.

- Klastry przechowujące dane (ang. storage clusters) - przykładami są OpenGFS, czy Lustre. Węzły takiego systemu klastrowego umożliwiają równoległy i szybki dostęp do danych przechowywanych w systemach plików tak, aby cały system udostępniał jeden, spójny z punktu widzenia użytkownika system plików.

- Klastry bazodanowe (ang. database clusters) - przykładem takiego systemu jest Oracle Parallel Server. Zadaniem węzłów klastra jest udostępnianie spójnego widoku bazy danych oraz umożliwienie szybkiego i niezawodnego dostępu do tych danych.

- Klastry dużej dostępności (ang. high-availability or failover clusters) - przykładami są ServiceGuard, czy Lifekeeper. Zasoby tworzące klaster są monitorowane i w przypadku wystąpienia błędu, w najgorszym wypadku wyłączenia się węzła, uruchamiane są specjalne skrypty mające na celu przeniesienie wykonywanych zadań na inny, działający węzeł klastra.

Oczywiście, część rzeczywistych rozwiązań próbuje łączyć wyżej wymienione typy klastrów tak, aby stworzyć jeden szybki, niezawodny i uniwersalny system. Dobrym i często stosowanym pomysłem jest połączenie cech klastra bazodanowego, klastra wysokiej dostępności oraz klastra przechowującego dane. Takie połączenie wydaje się wręcz oczywiste. Celem zespołów projektowych, zajmujących się rozwojem klastrów, jest stworzenie uniwersalnego klastra, łączącego w sobie możliwie najwięcej z wyżej wymienionych cech.

3.3 SSI - Single System Image

Słowem kluczowym, a raczej skrótem, stanowiącym kwintesencję rozwiązań klastrowych jest SSI (ang. Single System Image), co znaczy ”jednolity obraz systemu”. SSI jest mechanizmem dającym złudzenie jednolitego zasobu, tj. dysku, pamięci, czy procesora, podczas gdy faktycznie zasoby owe są rozproszone. Generalnie wyróżnia się 6 podejść do rozwiązań SSI:

- Klastry bazodanowe - Jednolity obraz systemu ma tutaj odniesienie do bazy danych. Instancje serwera bazy danych są uruchomione na każdym węźle systemu. Współpracują one jednak tak, aby umożliwić dostęp do tych samych danych, przechowywanych na różnych węzłach, w spójny sposób.

- Klastry przechowujące dane - Celem jednolitego obrazu systemu jest w tym wypadku umożliwienie przezroczystego dostępu do rozproszonych urządzeń (np. dysków) oraz działających na nich systemów plików. Użytkownik widzi jeden, wspólny system plików.

- Klastry udostępniające serwisy WWW - Serwer WWW jest widoczny z zewnątrz jako jeden serwer działający pod konkretnym adresem IP oraz na konkretnym porcie. Zapytania do serwera są przez system klastrowy dystrybuowane do węzłów tak, aby obciążenie całego systemu było równomierne.

- Klastry wysokiej wydajności - Jednolity obraz systemu jest realizowany z punktu widzenia wykonywania procesów. Nie ma tu jednolitego obrazu systemu plików, urządzeń, czy mechanizmów komunikacji międzyprocesowej. Równoległość obliczeń w klastrach tego typu jest zapewniona przez technologię ’bproc’ umożliwiającą migrację procesów. System klastrowy tego typu posiada tzw. główny węzeł, na którym inicjowane są wszystkie procesy. Są one następnie migrowane na inne węzły klastra za pomocą wywołań rexec(), czy rfork(). Nawet po przeniesieniu procesy widoczne są na głównym węźle tak, jakby były wykonywane lokalnie. Jednolity obraz systemu jest więc realizowany na węźle głównym.

- Klastry równoważące obciążenia - Paradygmat SSI jest w tym wypadku podobnie zrealizowany jak w klastrach wysokiej wydajności. Tym razem jednak, każdy węzeł ma możliwość inicjowania procesów i migrowania ich na pozostałe komputery klastra. Nie ma w tym wypadku 'węzła głównego' - jest natomiast 'węzeł źródłowy' każdego uruchomionego procesu. Proces przeniesiony na inny węzeł działa tak, jakby wciąż był wykonywany na węźle źródłowym. Widzi system plików, inne procesy, czy urządzenia przypisane do węzła źródłowego, a nie tego, na którym jest wykonywany. Podobnie, inne procesy uruchomione na tym samym węźle źródłowym widzą przeniesiony proces tak, jakby był wykonywany lokalnie. Użytkownik zalogowany do danego węzła widzi tylko procesy uruchomione na danym węźle i, co więcej, widzi je lokalnie - nie wie gdzie są realizowane.
- Idealny system klastrowy SSI - Ambicją twórców takiego systemu jest zintegrowanie cech innych systemów w jedno, możliwie uniwersalne rozwiązanie. Z założenia idealny system SSI powinien zapewniać:
  - Zunifikowany dla całego systemu klastrowego model procesu z niezawodnym mechanizmem równoważenia obciążeń pomiędzy węzłami klastra.
  - Zunifikowane modele urządzeń, a szczególności dysków i systemów plików z równoległym dostępem, umożliwiające implementację rozproszonych systemów bazodanowych.
  - Zunifikowany model sieciowy, umożliwiający zarówno niezawodny dostęp do klastra z zewnątrz, jak i z wewnętrznych węzłów, przy zachowaniu paradygmatu jednolitego obrazu systemu.
  - Zunifikowane modele obiektów służących do komunikacji międzyprocesowej IPC (ang. Inter Process Communication) takich, jak potoki, kolejki FIFO, semafory, systemy kolejkowania komunikatów, współdzielona pamięć, gniazda, czy sygnały. Wszystko to ma na celu umożliwić użytkownikom, programom i procesom współpracę bez względu na to, na których węzłach wykonywane są procesy.
  - Zunifikowany, pojedynczy system zarządzania całym systemem klastrowym, umożliwiający administrację zasobami wszystkich węzłów z dowolnego węzła klastra.
  - Silny mechanizm zapewniający niezawodność i wysoką dostępność wszystkich zasobów systemu, tj. elementów i usług sieciowych, urządzeń magazynowania danych, systemów plików, programów, procesów, czy mechanizmów IPC.

W dobie współczesnych, nowoczesnych rozwiązań klastrowych "jednolity obraz systemuódnosi się więc do większości lub nawet wszystkich zasobów komputerów tworzących klaster.

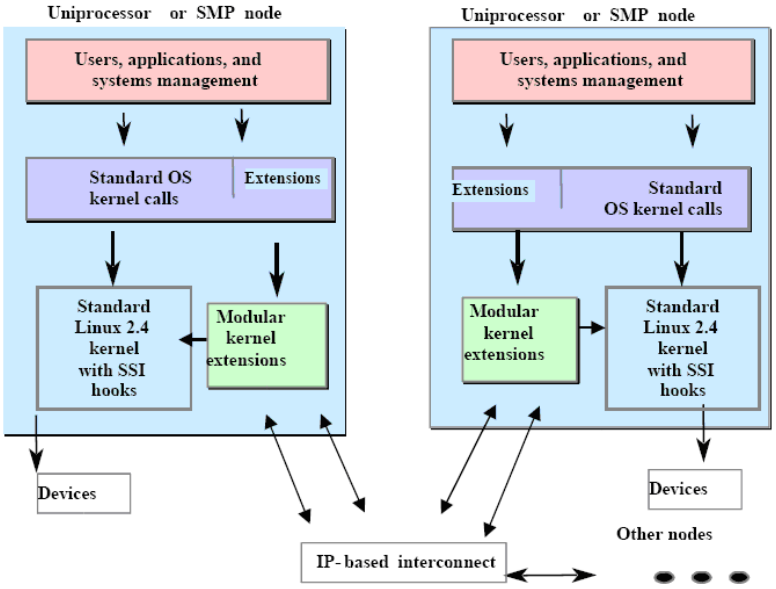
3.4 OpenSSI

OpenSSI jest wciąż rozwijanym projektem o otwartych źródłach, którego rezultatem ma być stworzenie klastrowego systemu operacyjnego działającego w oparciu o jądro systemu Linux. Celem zespołu projektowego OpenSSI jest stworzenie systemu realizującego jednolity obraz systemu (SSI) dla wszystkich zasobów i składników klastra. Aby to osiągnąć projektanci systemu zdecydowali się dokonać podziału klastra na składowe funkcjonalne, tzw. poziomy współpracy

lub bloki. Ścisłe zdefiniowanie sposobu działania systemu, a w szczególności komunikacji międzywęzłowej, na każdym poziomie oddzielnie stanowi o oryginalności rozwiązania i umożliwia integrację istniejących rozwiązań o otwartych źródłach w modularną całość i dopisanie brakujących składników tak, aby system był skalowalny i niezwykle wydajny. Zespół projektowy zdecydował się zapożyczyć część działających rozwiązań, które niejednokrotnie były wykorzystywane w innych Linux'owych, czy Unix'owych projektach klastrowych, często dedykowanych konkretnym zastosowaniom. Podejście modularne ma na celu umożliwienie wykorzystania sprawdzonych rozwiązań oraz podmianę poszczególnych składników systemu w przypadku stworzenia ich nowszych, lepszych odpowiedników. Umożliwia także rozwój systemu na poszczególnych poziomach niezależnie. Aktualną wersją systemu jest OpenSSI 1.9, przy czym w laboratorium RSO zainstalowana jest wersja 1.2.2. OpenSSI posiada implementacje działające pod różnymi systemami rodziny Linux, m.in. Debian, Red Hat, czy Knoppix, i jest udostępniane pod dwoma postaciami - jako źródła oraz jako pliki RPM.

3.5 Architektura rozwiązania OpenSSI

Klaster SSI działa w oparciu o pojedynczy, wspólny system plików. Każdy węzeł posiada niezależny system operacyjny, którego jądro potrafi komunikować się z systemami innych węzłów. Z założenia system plików nie powinien posiadać żadnych specyficznych dla danego węzła plików. Węzły systemu klastrowego zostają podłączone do klastra na wczesnym etapie uruchamiania



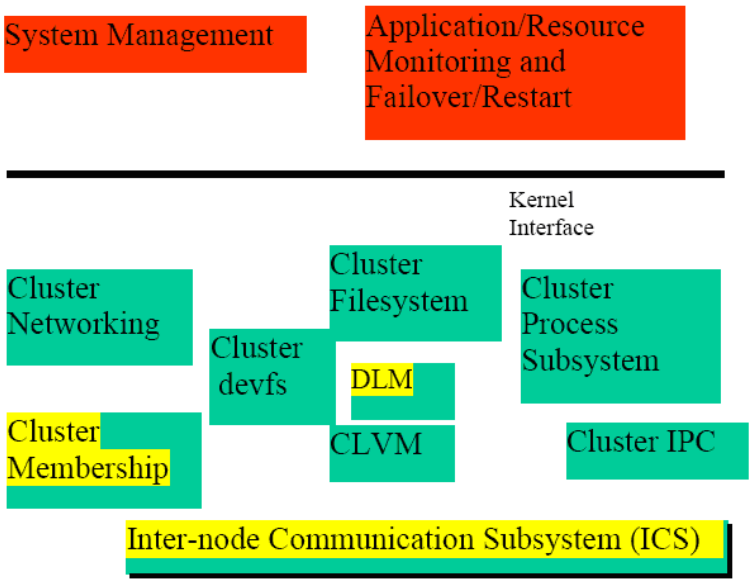
Rysunek 3: Komunikacja międzywęzłowa

systemu i "mają świadomość"bycia częścią systemu klastrowego. Z zewnątrz system SSI stanowi integralną całość już na poziomie poleceń systemowych. Współpracujące jądra zapewniają wspólną przestrzeń nazw dla wszelkich działających obiektów i umożliwiają spójny dostęp do zasobów. Aby system klastrowy mógł działać, jądra posiadają rozszerzenia umożliwiające koordynowanie dostępu do procesów, plików, czy urządzeń.

Elementy systemu SSI można podzielić na trzy kategorie:

- Elementy poza jądrem systemu - umożliwiają one zarządzanie systemem oraz jego wysoką dostępność. Do nich zalicza się system zarządzania klastrem, system monitorowania aplikacji i zasobów, oraz system odpowiedzialny za przywracanie i restart usług w wypadku wystąpienia błędu.
- Nowe składniki jądra - umożliwiają one obsługę dodatkowej, charakterystycznej dla systemu klastrowego funkcjonalności. Zalicza się do nich obsługę przynależności do klastra, system komunikacji międzywęzłowej, oraz rozproszony system blokad.
- Rozszerzenia istniejących fragmentów jądra - ich celem jest zapewnienie jednolitego obrazu systemu już na poziomie jądra. Do nich zalicza się systemy plików, zarządzanie procesami, obsługę sieci, czy obiekty komunikacji międzyprocesowej.

Poniższy diagram przedstawia lokalizację poszczególnych elementów składowych systemu klastrowego względem jego jądra. Następnie każdy z elementów jest opisany. Należy podkreślić, że



Rysunek 4: Elementy składowe systemu OpenSSI

elementy których działanie może mieć wpływ na realizację projektu rozproszonej bazy danych są opisane szerzej. Część elementów, które nie mają większego znaczenia dla realizacji projektu jest opisana na końcu, w podrozdziale ”Pozostałe podsystemy”.

3.6 Cluster Memebership

Podsystem przynależności do klastra dba o budowę klastra podczas uruchamiania poszczególnych węzłów oraz monitoruje obecność węzłów w klastrze. Jest również odpowiedzialny za dostarczanie informacji do pracujących węzłów, gdy któryś z nich odłączy się lub dołączy do klastra. Kod

serwisu przynależność do klastra (ang. Luster Membership Service - CLMS) jest realizowany na wszystkich węzłach. Węzły pracują w architekturze master-slave. W danym momencie działa tylko jeden węzeł główny. W sytuacji awarii, dowolny inny węzeł przejmuje jego funkcje. Z punktu widzenia węzła głównego każdy inny węzeł może być w jednym z następujących stanów: NeverUP, ComingUP, UP, Shutdown, Goingdown oraz Down. Przejścia między stanami są atomowe i przezroczyste, tj. widoczne w całym klastrze. Węzeł główny serwisu CLMS podejmuje decyzje o przynależności pozostałych węzłów, m.in. przeprowadza tak zwaną procedurę STONITH (ang. Shoot The Other Node In The Head), zapewniającą, że wykluczony z klastra węzeł nie będzie wykonywał żadnych operacji dyskowych. Rolą podsystemu CLMS jest więc przechowywanie aktualnego stanu każdego węzła i udostępnianie tej wiedzy innym węzłom tak, aby w całym klastrze istniał spójny widok systemu z punktu widzenia przynależności. Dodatkowo, system CLMS dostarcza interfejsów programistycznych, których celem jest udostępnianie informacji o przynależności, obecności i stanie węzłów w klastrze użytkownikom oraz programistom pracującym w środowisku OpenSSI. Tak, jak pozostałe podsystemy klastra OpenSSI, przy wymianie informacji pomiędzy węzłami CLMS polega na podsystemie komunikacji międzywęzłowej, który jest opisany w następnej części opracowania. Kod podsystemu CLMS został zapożyczony z projektu CI - Cluster Infrastucture for Linux (http://ci-linux.sourceforge.net).
Zaleca się wykonać następujące, proste ćwiczenie, którego celem jest zobrazowanie działania podsystemu przynależności do klastra, działającego w systemie OpenSSI. Kolejne kroki ćwiczenia wyglądają następująco:

- ”Po zalogowaniu do systemu - załóżmy, że użytkownik zalogował się do węzła nr 1 - wywołać następujące polecenie:

  #cluster -v

- Zalogować się na inny węzeł systemu - załóżmy, że jest to węzeł nr 2 - i ponownie wywołać powyższe polecenie. Rezultat zwróci na ekran informację o aktualnym stanie węzłów pracujących w systemie. W obu przypadkach wynik będzie identyczny

- Aby sprawdzić numer węzła, na którym zalogowany jest użytkownik, wywołać następujące polecenie:

  #clusternode_num

Powyżej opisane zostały dwa wybrane polecenia. Reszta przydatnych poleceń dostępnych dla zwykłego uzytkownik systemu opisana jest w dalszej cześci opracowania.

3.7 Internode Communication Subsystem - ICS

Podsystem komunikacji międzywęzłowej ICS jest systemem operującym na poziomie jąder. Za jego pośrednictwem większość pozostałych podsystemów ma możliwość realizowania komunikacji międzywęzłowej. System ICS zbudowany jest z dwóch warstw - warstwy górnej, niezależnej od transportu, oraz warstwy dolnej, zależnej od transportu. Warstwa górna posiada interfejs wykorzystywany przez inne podsystemy jądra i za jego pomocą udostępnia trzy mechanizmy komunikacji: RPC, asynchroniczne RPC oraz przekazywanie komunikatów. Warstwa górna wyposażona jest w dynamiczną pulę wątków, służących do obsługi przychodzących zapytań, które

bardzo często mają określone priorytety i muszą być kolejkowane tak ,aby nie powodowały zakleszczeń. Warstwa dolna posiada interfejs do niezawodnych mechanizmów transportowych partych o określone rozwiązania sprzętowe. Obecna implementacja systemu OpenSSI wspiera transport TCP. Podsystem komunikacji międzywęzłowej ściśle współpracuje z systemem CLMS (ang. Cluster Membership Service) tak, aby zapewnić komunikację z każdym węzłem, który włącza się do klastra oraz, aby usunąć cały ruch do i od węzła, który opuścił klaster. Podobnie, jak CLMS, kod podsystemu ICS został zapożyczony z projektu CI - Cluster Infrastucture for Linux (http://ci-linux.sourceforge.net).

3.8 Clusterwide Filesystem

Systemem plików, specjalnie stworzonym dla projektu OpenSSI, jest CFS. Celem jego działania jest automatyczne, przezroczyste i spójne przedstawienie hierarchii katalogów na wszystkich węzłach klastra. Udostępnia on identyczny widok hierarchii katalogów, bez względu na to, na którym węźle pracuje użytkownik. System CFS charakteryzuje się następującymi czynnikami:

- Jest wirtualnym systemem plików, działającym w oparciu o fizyczny system plików taki, jak ext3, czy XFS.
- Węzły klastra zorganizowane są w architekturę klient-serwer, tzn. jest jeden główny węzeł, który udostępnia wspólny widok hierarchii katalogów (ang. root filesystem), a pozostałe węzły odwołują się do niego. Istotną cechą systemu CFS jest to, że można w głównym systemie plików montować inne, charakterystyczne dla pozostałych węzłów tak, aby udostępnić dyski i partycje owych węzłów. Jest to niezwykle istotna cecha z punktu widzenia projektantów rozproszonego systemu bazodanowego. Zamontowanie partycji dysków poszczególnych węzłów umożliwia instalację instancji serwera na poszczególnych węzłach, celem wprowadzenia redundancji.
- Hierarchia katalogów tworzona przez węzeł główny jest automatycznie dystrybuowana do innych węzłów.
- System jest odporny na awarię węzła, który w danym momencie pełni rolę serwera systemu plików. Dowolny inny węzeł, mający fizyczny dostęp do urządzania dyskowego, może automatycznie przejąć rolę serwera w sytuacji awaryjnej
- CFS wspiera obsługę innych klastrowych systemów plików takich, jak OpenGFS (http://opengfs.sourcef czy Lustre (http://www.Lustre.org).
- System jest w pełni spójny (ang. coherent), co oznacza, że każdy węzeł widzi zmianę pliku, czy struktury katalogu, gdy tylko ona nastąpi.
- System CFS traktuje wszystkie węzły identycznie, co oznacza, że zarówno serwer systemu plików, jak i klienty mają dostęp do zawartości systemu plików poprzez te same mechanizmy.

Podsumowując warto podkreślić, że klastrowy system plików CFS doskonale nadaje się do realizacji klastrowej bazy danych. Udostępnia bowiem spójny widok głównego systemu plików,

jak również systemów poszczególnych węzłów. Można więc wprowadzić redundancję do bazy danych poprzez zainstalowanie niezależnych serwerów wraz z repozytoriami na poszczególnych węzłach i mieć do nich dostęp z poziomu pozostałych węzłów systemu OpenSSI.
Zaleca się wykonać następujące, proste ćwiczenie, którego celem jest zobrazowanie działania klastrowego systemu plików CFS, działającego w systemie OpenSSI. Kolejne kroki ćwiczenia wyglądają następująco:

- Uruchomić program vim po zalogowaniu do systemu - załóżmy, że użytkownik zalogował się do węzła nr 1.
- Wprowadzić w edytorze kilka linijek tekstu.
- Zapisać plik, ale nie wyłączać edytora.
- Zalogować się do innego węzła systemu - załóżmy, że jest to węzeł nr 2 - z prawami do odczytu wcześniej zapisanego pliku - może to być ten sam użytkownik.
- Wypisać zawartość pliku na ekran, wywołując następujące polecenie:

  #cat cieka_do_pliku

W wyniku wywołania polecenia cat użytkownik będzie mógł odczytać tekst zapisany w pliku. Oznacza to, że widok systemu plików jest spójny w obrębie klastra. Bez względu na to, czy tworzymy, przemieszczamy, edytujemy, czy też usuwamy pliki lub katalogi na jednym węźle, wszystkie zmiany są natychmiast dostrzegalne z poziomu pozostałych węzłów systemu.

3.9 Clusterwide Process Subsystem

Podsystem zarządzania procesami, zastosowany w systemie OpenSSI sprawia, że wszystkie procesy, na wszystkich węzłach, są przezroczyście widoczne i dostępne dla innych procesów. Z punktu widzenia użytkownika sprawiają wrażenie, jakby były realizowane na lokalnej maszynie. Podsystem umożliwia przezroczyste rozpraszanie grup oraz rodzin procesów, a także procesów należących do jednej sesji, procesu debugującego i debugowanego, jak również par procesów: wysyłającego i odbierającego sygnał. Użytkownik ma dostęp do jednolitego dla całego klastra widoku systemu /proc na każdym węźle. Podsystem zarządzania procesami umożliwia także migrację procesu z węzła na węzeł w trakcie jego wykonywania, zarówno ręczną, z pełną kontrolą nad procesem migracji, jak również automatyczną i w pełni przezroczystą. Mechanizmy migracji pozwalają na realizację elastycznego podsystemu równoważenia obciążeń węzłów. System gwarantuje również poprawną obsługę procesów w przypadku odłączania się i powtórnego dołączania do klastra poszczególnych węzłów. Opisane cele podsystemu zarządzania procesami zostały osiągnięte dzięki następującym czynnikom:

- Każdy proces działający w obrębie klastra ma przydzielony unikalny dla całego systemu klastrowego numer identyfikacyjny - PID. Numer ten pozostaje niezmienny w czasie działania procesu, nawet wówczas, gdy proces jest migrowany z węzła na węzeł.

- Informacje o relacjach miedzy procesami są przechowywane w strukturze ”vproc- wirtualnej strukturze stworzonej po to, aby umożliwić wykorzystanie tradycyjnych mechanizmów obsługi procesów w systemach Linux - struktur ”task_struct”. Dla jednego procesu może być klika struktur ”vproc”przechowywanych na różnych węzłach i reprezentujących relacje z innymi procesami, w jakich dany proces się znajduje. Poszczególne struktury ”vproc”wskazują na strukturę ”task_struck”, która przechowywana jest w węźle, w którym proces został powołany do życia - pozwala to na zachowanie kompatybilności. Typowe wywołania systemowe czy funkcje jądra, które nie są związane z mechanizmami klastra odwołują się bezpośrednio do struktury ”task_struct”, pozostałe natomiast odwołują się do struktury ”vproc”, która z kolei odwołuje się do struktury ”task_struct”.

- Wywołania systemowe (ang. system calls) realizowane są na węźle, na którym w danym momencie jest wykonywany proces. Jeżeli istniej potrzeba odwołania się do zdalnych zasobów, aby zrealizować wywołanie systemowe, odpowiednie odwołania realizowane są przezroczyście za pomocą podsystemu komunikacji międzywęzłowej (ICS).

- System klastrowy przechowuje redundantne informacje o przebiegu danego procesu i relacjach z innymi procesami. Umożliwia to rekonstrukcję stanu procesu i jego relacji po upadku dowolnego węzła.

- Aby umożliwić kontrolowaną migrację procesów przez aplikacje świadome istnienia klastra, wprowadzono kilka użytecznych funkcji: Rexec(), rfork(), migrate(), oraz kill3(). Dodano również sygnał SIGMIGRATE, który pomaga zarządzać migracją pojedynczych procesów, jak również grup.

- Istnieje możliwość skonfigurowania opcjonalnego podsystemu równoważenia obciążeń klastra, który pozwala na przezroczysta migrację procesów tak, aby optymalnie wykorzystać możliwości obliczeniowe poszczególnych węzłów.

Podsumowując można powiedzieć, że podsystem zarządzania procesami w systemie OpenSSI udostępnia spójny widok wszystkich procesów realizowanych w obrębie klastra, zarówno w odniesieniu do wywołań systemowych, jak i z punktu widzenia systemu /proc/¡pid¿. Pozwala więc na stosowanie nieświadomych istnienia klastra poleceń, jak ps czy gdb. Podsystem umożliwia również łatwą i kontrolowaną, jak również w pełni przezroczystą migrację procesów, nawet w trakcie ich wykonywania.
Zaleca się wykonać następujące, proste ćwiczenie, którego celem jest zobrazowanie idei przezroczystości lokalizacji procesów w systemie OpenSSI. Kolejne kroki ćwiczenia wyglądają następująco:

- Uruchomić program vim po zalogowaniu do systemu - załóżmy, że użytkownik zalogował się do węzła nr 1.

- Zalogować się do innego węzła systemu - załóżmy, że jest to węzeł nr 2.

- Sprawdzić ogólno-klastrowy numer procesu vim z konsoli drugiego węzła:

  #ps grep | vim

W tym miejscu należy nadmienić, iż polecenie ps jest niezmienione, tzn. jest identyczne, jak w tradycyjnym systemie Linux, a więc nie jest świadome istnienia klastra. Mimo to, za jego pomocą można uzyskać nr procesu uruchomionego na innym węźle (w tym wypadku na węźle nr 1). Dzieje się tak, ponieważ jądro OpenSSI ma przezroczysty dostęp do zdalnych obiektów. Z punktu widzenia użytkownika wydaje się, że proces jest uruchomiony na węźle, na którym wywołane zostało polecenie ps.

- ”Upewnić się, gdzie faktycznie jest wykonywany proces vim:

  #where_pid nr_procesu_zwrd̆ż˝ony_przez_ps

- Źakończyć proces vim poprzez wywołanie polecenia kill z konsoli węzła nr 2:

  #kill nr_procesu_zwrd̆ż˝ony_przez_ps

Podobnie, jak ps, polecenie kill nie jest świadome istnienia klastra. Mimo to, można za jego pomocą zakończyć proces, który faktycznie jest realizowany na innym węźle systemu. Tak więc, podsystem zarządzania procesami nie tylko umożliwia przezroczysty widok procesów w obrębie klastra, ale pozwala wysyłać do nich sygnały.

3.10 Clusterwide Inter-Process Communication - IPC

System OpenSSI udostępnia użytkownikom (programistom) wiele rodzajów komunikacji międzyprocesowej IPC. Wyróżnia się, między innymi, sygnały, potoki, kolejki FIFO, semafory, systemy kolejkowania komunikatów, współdzieloną pamięć, gniazda, itd. Architektura IPC została w systemie OpenSSI zaimplementowana niezwykle przejrzyście. Klaster udostępnia jedną, spójną przestrzeń nazw dla wszelkich obiektów komunikacji międzyprocesowej, tworzonych na wszystkich węzłach działających w systemie. Klaster jest przystosowany do obsługi standardowych nazw obiektów IPC. Za zarządzanie przestrzenią nazw obiektów odpowiedzialny jest serwer nazw, który jest automatycznie reaktywowany po ewentualnej awarii węzła, na którym do tej pory pracował. - nie stanowi on więc słabego punktu całej koncepcji. Każdy obiekt komunikacji IPC jest tworzony lokalnie, na tym węźle, na którym wykonywany jest proces, który powołuje go do życia. W przeciwieństwie do procesów, obiekty IPC nie mogą być jednak przenoszone między węzłami. Gdy zostaną utworzone na danym węźle, muszą na nim pozostać do momentu ich usunięcia. Niemniej jednak, procesy wykonywane na innych węzłach, niż obiekty IPC, mogą odwoływać się do tych obiektów tak, jakby znajdowały się na tym samym węźle. Tak więc, komunikacja między procesami wykonywanymi na różnych węzłach jest w pełni przezroczysta. Opisane zalety podsystemu komunikacji międzyprocesowej zostały osiągnięte dzięki odpowiedniej implementacji serwera nazw, który dodatkowo cechuje się następującymi czynnikami:

- System IPC jest obsługiwany przez pojedynczy serwer nazw, dzięki czemu każdorazowe odwołanie się do istniejącego obiektu IPC, czy utworzenie nowego obiektu, oznacza pojedyncze wywołanie RPC.

- Serwer nazw jest odporny na awarię węzła, na którym pracuje, dzięki mechanizmom zapewniającym pełną jego odbudowę na dowolnym innym węźle klastra.

Aby przyspieszyć działanie systemu, projektanci OpenSSI zdecydowali się umożliwić gromadzenie informacji o aktualnie działających obiektach IPC w pamięciach podręcznych poszczególnych węzłów, których zawartość w razie potrzeby jest odświeżana poprzez odwołania do serwera nazw. Zaleca się wykonać następujące, proste ćwiczenie, którego celem jest zobrazowanie idei przezroczystości podsystemu IPC w systemie OpenSSI. Kolejne kroki ćwiczenia wyglądają następująco:

- Po zalogowaniu do systemu - załóżmy, że użytkownik zalogował się do węzła nr 1 - wywołać następujące polecenie:

  #mkfifo /tmp/fifo

- Zalogować się do innego węzeł systemu - załóżmy, że jest to węzeł nr 2 - i wywołać następujące polecenie:

  #echo 'Nie ma to jak pisać projekty :)' > /tmp/fifo

- "Wywołać poniższe polecenie z konsoli węzła nr 1:

  #cat /tmp/fifo

W wyniku wywołania ostatniego polecenia użytkownik powinien otrzymać tekst, który wcześniej został przekierowany do kolejki. Programy mkfifo, echo oraz cat są nieświadome istnienia klastra i są to dokładnie takie same polecenia, jakie stosuje się w zwykłym systemie rodziny Linux. System OpenSSI wspiera przezroczyste działanie mechanizmów IPC, dlatego też kolejka utworzona z konsoli węzła pierwszego jest widoczna i dostępna na węźle drugim. Dokładnie tak samo obsługiwane są wszelkie inne metody komunikacji międzyprocesowej.

3.11 Pozostałe podsystemy

Pozostałe podsystemy działające w systemie OpenSSI mają mniejsze znaczenie z punktu widzenia realizacji rozproszonej bazy danych. Poniżej znajdują się krótkie opisy ich działania.

- Rozproszony system blokad (ang. Distributed Lock Manager) - Pracuje na każdym węźle i dba o spójność wszelkich rodzajów pamieci podręcznych zaimplementowanych w systemie OpenSSI. Funkcjonalność tego podsystemu została zaimplementowana na podstawie rozwiązania o otwartych źródłach firmy IBM.

- Obsługa urządzeń (ang. Clusterwide Devices Subsystem) - Zapewnia spójny sposób nazywania i montowania urządzeń na węzłach pracujących w obrębie klastra tak ,aby były one widoczne i dostępne z dowolnego miejsca systemu.

- Obsługa sieci (ang. Clusterwide Networking Subsystem) - Udostępnia jeden, wspólny adres IP dla klastra, widoczny od strony zewnętrznej, tj. spoza klastra, oraz rozdziela sesje TCP inicjowane z zewnątrz, tak aby urządzenia były optymalnie obciążone. Funkcjonalność tego podsystemu została zaimplementowana na podstawie projektu o nazwie Linux Virtual Server (http://www.linuxvirtualserver.org).

- System zarządzania (ang. Clusterwide System Management) - Podsystem służy do obsługi klastra OpenSSI przy użyciu komend zbliżonych do typowych komend stosowanych w systemach Linux. Dodatkowo, podsystem zarządzania umożliwia bardzo prosty sposób dołączania nowych węzłów do klastra. Funkcjonalność tego podsystemu została zaimplementowana na podstawie projektu o nazwie NonStop Clusters for Unixware - NSC.
- Obsługa awarii (ang. Application/Resource Monitoring and Failover/Restart) - Podsystem pozwala definiować reakcje systemu klastra w przypadku wystąpienia sytuacji awaryjnych jak, na przykład, odłączenie węzła, czy zatrzymanie aplikacji.

3.12 Przydatne polecenia

Polecenie cluster jest podstawowym poleceniem służącym do podstawowej obsługi klastra - wypisuje ono informacje na temat węzłów, które aktualnie pracują w klastrze. Bez żadnych opcji polecenie wypisuje informacje o wszystkich węzłach. Polecenie posiada kilka opcji dodatkowych:

- cluster -v -> Wypisuje stan węzłów w klastrze.
- cluster -V -> Wypisuje szczegółowe informacje o każdym węźle.
- cluster -m -> Wypisuje maksymalną liczbę węzłów wspieraną przez klaster.
- cluster -n -> Wypisuje liczbę w pełni podniesionych węzłów klastra.
- cluster -r -> WWypisuje aktualnie zainstalowaną wersję systemu OpenSSI.

Użytkownik ma również dostęp do następujących poleceń.

- clusternode_avail -n nr_węzła -> Wypisuje status węzła o danym numerze.
- clusternode_num -> Wypisuje numer danego węzła.

Poleceniem loads użytkownik może sprawdzić obciążenie każdego z węzłów aktualnie aktywnych w klastrze.
Jest jeszcze wiele przydatnych poleceń, niestety zwykły użytkownik systemu nie ma do nich praw. Niewykluczone, że przy realizacji projektu rozproszonej bazy danych mogą się przydać. Ich składnia oraz dokładne opisy znajdują się na stronie http://ci-linux.sourceforge.net/docs.shtml. Wart przeczytania jest również dokument Pt. Ńiezawodne usługi w rozwiązaniach SSI"Karola Ostrowskiego, dostępny na stronie http://www.ia.pw.edu.pl/ tkruk/. Porusza on aspekty niezawodnościowe przy realizacji projektów informatycznych w oparciu o system OpenSSI.

4 Instalacja serwera MySQL na koncie użytkownika

4.1 Wprowadzenie do MySQL

System MySQL powstał w połowie lat dziewięćdziesiątych. Jego autorem był Michael ”Monty” Widenius ze szwedzkiej firmy TcX DataKonsult AB. Pierwsza powszechnie dostępna wersja została wydana w maju 1995 roku. Obecnie rozwojem MySQL zajmuje się firma MySQL AB.

4.2 System Zarządzania Bazą Danych

System Zarządzania Bazą Danych, SZBD (ang. *Data Base Management System - DBMS*) nazywany też serwerem baz danych lub systemem baz danych to oprogramowanie bądź system informatyczny umożliwiający dostęp i zarządzanie jedną lub więcej bazami danych. DBMS jest powłoką, która otacza bazę danych i za pomocą której dokonują się na niej wszystkie operacje. Kopalnią wiedzy na temat baz danych i systemów zarządzania nimi może być książka [EN05].

4.3 MySQL

MySQL to obok PostgreSQL najpopularniejszy z wolnodostępnych silników relacyjnych baz danych [wik]. Jest to przykład SZDB bazującego na architekturze klient-serwer. Książki [Atk03, MS02] zawierają bardzo dużą dawkę wiedzy na temat MySQL. Przygotowując się do realizacji tej fazy projektu w większej mierze się na niej opierano.

4.4 Licencja MySQL

MySQL jest oprogramowaniem *open source*. Jego kod źródłowy jest ogólnie dostępny. Licencja MySQL-a (*GPL – General Public License*) pozwala na jego używanie bez ograniczeń. Jeśli jednak zechce się wbudować MySQL w swój komercyjny produkt, potrzebna jest osobna licencja.

4.5 Przygotowanie paczki

Instalację MySQL można rozpocząć od kodu źródłowego lub skompilowanych plików binarnych. w obu przypadkach należy odwiedzić stronę domową MySQL [mysa] i pobrać odpowiednie archiwa. Wykorzystanie plików binarnych jest wygodnym rozwiązaniem. Istnieją archiwa tych plików dla wielu systemów operacyjnych. Standardowe instalacje opisane są w [Atk03, MS02].

Zadanie projektowe wymagało jednak instalacji bez konieczności angażowania katalogów systemowych i konta `root`, należało więc przygotować odpowiednią paczkę.

4.6 Wersja

Zespół, który nadzoruje MySQL stara się jednocześnie stymulować rozwój bazy danych oraz utrzymywać jej stabilne wersje [Atk03]. Należy generalnie wybierać najnowszą dostępną wersję. Takie też były zalecenia prowadzącego przedmiot. Pobrano więc źródła wersji 5.1.7-beta. W przypadku, gdyby pojawiła się nowa wersja należy już jednak pozostać przy wersji z którą rozpoczęto prace projektowe.

4.7 Kompilacja źródeł

Do poprawnego działania paczki potrzebna jest kompilacja źródeł z odpowiednimi parametrami, takimi jak inny port uruchomienia usługi i prawa użytkownika do uruchomienia demona serwera MySQL. Nastepnie konieczna jest modyfikacja plików konfiguracyjnych.

Najpierw dokonano kompilacji kodu źródłowego na jednym z kont członków zespołu (`kfabjans`), na `openone.ia.pw.edu.pl`. Kod źródłowy przygotowany jest za pomocą `autoconf`, wymagana więc była standardowa procedura:

```
$ configure
$ make
$ make install
```

Skompilowano paczkę tak, aby pliki umieszone zostały w katalogu `mysql` w domowym katalogu użytkownika. Jako domyślny port wybrano `2002`, ale zawsze będzie on mógł być modyfikowany według potrzeb użytkowników. Wystarczy do tego celu edycja pliku konfiguracyjnego (opisany jest on w następnym podrozdziale). Podsumowując, do konfiguracji użyto następujących opcji:

```
--prefix=$HOME/mysql
--exec-prefix=$HOME/mysql
--with-unix-socket-path=$HOME/mysql/thesocket
--with-tcp-port=2002
--with-innodb
```

4.8 Zmiany ścieżek dostępu

Ponieważ kompilacja odbyła sie na koncie użytkownika `kfabjans` należało zamienić w powstałej z kompilacji instalacji ciagi: `/home/rso/kfabjans/` na `$HOME`, tak aby instalując paczkę u dowolnego użytkownika katalog domowy był poprawny.

4.9 Przygotowanie skryptów startujących i zamykających usługi

Następnie stworzono skrypty, które będą startować i zamykać demona serwera oraz skrypt uruchamiający narzędzie `mysql`.

4.9.1 Serwer

Serwer MySQL to plik wykonywalny **mysqld**. Jest on uruchamiany na stałe i przyjmuje połączenia ze strony klientów.

Skrypt, któy będzie uruchamiał serwer wygląda następująco:

```
#!/bin/sh

if test -s $HOME/mysql/the.pid
then
    MY=1;
    PID=$(cat $HOME/mysql/the.pid);
```

```
else
    MY=0;
fi

case $# in
1)
    if test $1 = "start"
    then
        if test $MY -eq 1
        then
            echo 'MySQL process is already running...'
            exit 1
        fi

        if test $MY -eq 0
        then
            $HOME/mysql/libexec/mysqld &
            exit 1
        fi
    fi

    if test $1 = "stop"
    then
        if test $MY -eq 1
        then
            kill $PID
            echo 'MySQL process stopped properly...'
            exit 1
        else
            echo 'MySQL process already stopped'
            exit 1
        fi
    fi

    if test $1 = "restart"
    then
        if test $MY -eq 1
        then
            kill $PID
            $HOME/mysql/libexec/mysqld &
            exit 1
        else
            echo 'MySQL process already stopped'
            $HOME/mysql/libexec/mysqld &
            exit 1
```

```
        fi
    fi

    echo 'Wrong option, try one of the following ways'
    echo './mysql_server start'
    echo './mysql_server stop'
    echo './mysql_server restart'
    exit 1
    ;;
*)
    echo 'Improper number of arguments,
          try one of the following ways'
    echo './mysql_server start'
    echo './mysql_server stop'
    echo './mysql_server restart'
    exit 1
    ;;
esac
```

4.9.2 Klient

Najważniejszym programem dla użytkownika MySQL jest narzędzie **mysql**. Umożliwia ono formułowanie zapytań do bazy w trybie interaktywnym lub wsadowym [Atk03]. Skrypt, który będzie uruchamiał narzędzie klienta wygląda następująco:

```
#!/bin/sh

export LD_LIBRARY_PATH=$HOME/mysql/lib/mysql

$HOME/mysql/bin/mysql $*;
```

Jak widać, przed uruchomieniem konieczne jest wyeksportowanie ścieżki, według której szukane będą biblioteki na koncie użytkownika.

4.10 Główna część paczki

Utworzono archiwum `tar` ze skompilowanej instalacji oraz powyższych stworzonych skryptów i spakowano go, tworząc podstawową część paczki:

```
mysql_install.tar.gz
```

4.11 Przygotowanie skryptu instalacyjnego

Przygotowano skrypt instalacyjny, rozpakowujący i konfigurujący paczkę na koncie użytkownika. Instalacja będzie odbywała się w następujący sposób:

```
#!/bin/sh

if test $# -ne 0
then
    echo "Improper number of arguments";
    echo "Try install.sh without any arguments";
    exit 1
fi

if test -d $HOME/mysql
then
    echo "MySQL already installed...";
    exit 1
fi
#-----------------------------------------------------
```

Następuje rozpakowanie paczki:

```
#-----------------------------------------------------
PWD=$(pwd);
#extraction of tar content
if test $PWD = $HOME
then
    tar -xvvzf mysql.tar.gz
else
    tar -xvvzf mysql.tar.gz
    cp -R $PWD/mysql $HOME/mysql
    rm -R $PWD/mysql
fi

if test ! -d $HOME/mysql
then
    echo "Corrupted tar archive...";
    exit 1
fi
#-----------------------------------------------------
```

Konieczna jest modyfikacja praw dostępu dla głównego katalogu MySQL:

```
#-----------------------------------------------------
#modyfication of privileges
chgrp -R rso1 $HOME/mysql
chmod -R u+wrx $HOME/mysql
chmod -R g+wrx $HOME/mysql

#-----------------------------------------------------
```

Następnie tworzony jest plik konfiguracyjny. Standardowo, uruchomione narzędzie MySQL szuka w kilku miejscach określonych plików [Atk03]. Kolejno szukane i wykonywane są 3 pliki:

- `/etc/my.cnf` - plik o zakresie systemowym,
- `my.cnf` - w katalogu danych,
- `my.cnf` - w katalogu użytkownika.

Obecność żadnego z plików nie jest obowiązkowa, ale jeśli istnieją, to każdy następny nadpisuje wartości z poprzednio odczytanych. Naspisane zostaną nawet wartości nadane zmiennym środowiskowym.

Plik konfiguracyjny ma prostą strukturę Krzyżyk (#) oznacza komentarz. Opcje w pliku rozdzielane są nagłówkami, które zawarte są w nawiasach kwadratowych. Opcje zawarte pod nagłówkiem `[server]` dotyczą serwera. Opcje zawarte pod nagłówkiem `[client]` mają natomiast zastosowanie do wszystkich klientów. Nazwy pozostałych nagłówków pochodzą od nazw programów klienta.

W przygotowanej paczce będzie się instalował tylko plik `.my.cnf` w katalogu `$HOME`:

```
#-------------------------------------------------
#creating config file .my.cnf

echo "[client]
socket          = $HOME/mysql/thesocket
port            = 2002

[mysqld_safe]
socket          = $HOME/mysql/thesocket
err-log         = $HOME/mysql/the.log

[mysqld]
pid-file        = $HOME/mysql/the.pid
socket          = $HOME/mysql/thesocket
port            = 2002
basedir         = $HOME/mysql
datadir         = $HOME/mysql/lib/mysql
tmpdir          = /tmp
language        = $HOME/mysql/share/mysql/english
key_buffer              = 16M
max_allowed_packet      = 16M
thread_stack            = 128K
query_cache_limit       = 1048576
query_cache_size        = 16777216
query_cache_type        = 1

[mysqldump]
quick
```

```
quote-names
max_allowed_packet      = 16M" > $HOME/.my.cnf

if test ! -s $HOME/.my.cnf
then
    echo "Configuration file cannot be created..";
    exit 1
fi
#-------------------------------------------------
```

Konieczna jest kolejna modyfikacja praw dostępu – tym razem dla pliku konfiguracyjnego:

```
#-------------------------------------------------
chgrp rso1 $HOME/.my.cnf
chmod u+wrx $HOME/.my.cnf
chmod g+wrx $HOME/.my.cnf
#-------------------------------------------------
```

Następnie uruchamiany jest skrypt **mysql_install_db** [Atk03], który tworzy domyślne tabele w bazie `mysql`. Zwykle jest on uruchamiany raz podczas pierwszej instalacji MySQL. Jeżeli tabela uprawnień istnieje, skrypt nie wykonuje żadnych działań:

```
#-------------------------------------------------
#executing install_DB

$HOME/mysql/bin/mysql_install_db
#-------------------------------------------------
```

Należy utworzyć linki symboliczne do skryptów uruchamiających serwer i klienta i umieścić je w katalogu domowym:

```
#-------------------------------------------------
#creating symlinks and mysqld deamon service

ln -s $HOME/mysql/mysql_client $HOME/mysql_client

if test ! -s $HOME/mysql_client
then
    echo "MySQL client is not available";
    exit 1
fi
#-------------------------------------------------
```

Na koniec ostatnie modyfikacje praw dostępu:

```
#-------------------------------------------------
chgrp rso1 $HOME/mysql_client
chmod u+wrx $HOME/mysql_client
```

```
chmod g+wrx $HOME/mysql_client

ln -s $HOME/mysql/mysql_server $HOME/mysql_server

if test ! -s $HOME/mysql_server
then
    echo "MySQL server is not available";
    exit 1
fi

chgrp rso1 $HOME/mysql_server
chmod u+wrx $HOME/mysql_server
chmod g+wrx $HOME/mysql_server

echo "Installation complete..."
```

4.12 Przygotowanie skryptu dezinstalacyjnego

Przygotowano również skrypt dezinstalacyjny, usuwający `mysql` z konta użytkownika. Dezinstalacja będzie odbywała się w następujący sposób:

```
#!/bin/sh

#removing all files

if test ! -s $HOME/.my.cnf
then
    echo ".my.cnf was not removed because
          it does not exist";
fi

if test -s $HOME/.my.cnf
then
    rm $HOME/.my.cnf;
    echo "File .my.cnf was removed...";
fi

if test ! -s $HOME/mysql_client
then
    echo "mysql_client was not removed
          because it does not exist"
fi

if test -s $HOME/mysql_client
then
```

```
    rm $HOME/mysql_client
    echo "File mysql_client was removed...";
fi

if test ! -s $HOME/mysql_server
then
    echo "mysql_server was not removed
          because it does not exist"
fi

if test -s $HOME/mysql_server
then
    rm $HOME/mysql_server;
    echo "File mysql_server was removed...";
fi

#removing direcotry

if test ! -d $HOME/mysql
then
    echo "mysql directory was not removed
          because it does not exist"
fi

if test -d $HOME/mysql
then
    rm -fR $HOME/mysql;
    echo "Directory mysql was removed";
fi
```

4.13 Instalacja MySQL

Gotowa paczka jest spakowana i nazywa się:

`mysql_install.tar.gz`

Zawiera ona trzy główne pliki:

```
install.sh
uninstall.sh
mysql.tar.gz
```

Pliki należy rozpakować do dowolnego wspólnego katalogu, a następnie uruchomić skrypt `./install.sh`. MySQL docelowo instaluje się w `$HOME/mysql`.

4.14 Uruchomienie

4.15 Serwer

Po instalacji, usługę serwera MySQL należy uruchomić w następujący sposób:

```
$ ./mysql_server start
```

Serwer MySQL restartowany jest komendą:

```
$ ./mysql_server restart
```

Zatrzymanie następuje po wydaniu komendy:

```
$ ./mysql_server stop.
```

4.16 Klient

Aby uruchomić narzędzie `mysql` należy wydać polecenie:

```
$ mysql_client
```

4.17 Konfiguracja

Opisywany już plik `.my.cnf` generuje się w trakcie instalacji i jest indywidualny dla każdego konta. W pliku tym przechowywane są informacje odnośnie konfiguracji MySQL. Można go modyfikować wedle indywidualnych potrzeb, zmieniając np. port.

4.18 Przeprowadzone testy

Uruchomiono serwer i narzędzie klienta na koncie użytkownika `jnowacki` i przeprowadzono proste testy, pokazujące działanie zainstalowanego MySQL-a:

```
jnowacki@lab4-cs3:~$ ./mysql_server start

jnowacki@lab4-cs3:~$ 060402 16:41:13
InnoDB: Started; log sequence number 0 43655
060402 16:41:13 [Note] /home/rso/jnowacki/mysql/libexec/mysqld:
ready for connections.
Version: '5.1.7-beta'  socket: '/home/rso/jnowacki/mysql/thesocket'
port: 2002  Source distribution

jnowacki@lab4-cs3:~$ ./mysql_client
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.1.7-beta

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

```
mysql> status
--------------
/home/rso/jnowacki/mysql/bin/mysql
Ver 14.12 Distrib 5.1.7-beta, for pc-linux-gnu (i686)
using  EditLine wrapper

Connection id:          7
Current database:
Current user:           jnowacki@localhost
SSL:                    Not in use
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server version:         5.1.7-beta
Protocol version:       10
Connection:             Localhost via UNIX socket
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    latin1
Conn.  characterset:    latin1
UNIX socket:            /home/rso/jnowacki/mysql/thesocket
Uptime:                 20 min 23 sec

Threads: 1  Questions: 30  Slow queries: 0  Opens: 0
Flush tables: 1  Open tables: 8
Queries per second avg: 0.025
--------------


mysql> create table ksiazka (
     > tytul varchar(32),
     > autor varchar(64)
     > );
Query OK, 0 rows affected (0,01 sec)


mysql> insert into ksiazka values ('MySQL', 'Leon Atkinson');
Query OK, 1 row affected (0,00 sec)


mysql> select * from ksiazka;
+-------+---------------+
| tytul | autor         |
+-------+---------------+
```

```
| MySQL | Leon Atkinson |
+-------+---------------+
1 row in set (0,00 sec)


mysql> exit
Bye

jnowacki@lab4-cs3:~$ ./mysql_server stop
MySQL process stopped properly...
060402 17:20:02 [Note] /home/rso/jnowacki/mysql/libexec/mysqld:
Normal shutdown
060402 17:20:02  InnoDB: Starting shutdown...

jnowacki@lab4-cs3:~$ 060402 17:20:04
InnoDB: Shutdown completed; log sequence number 0 43655
060402 17:20:04 [Note] /home/rso/jnowacki/mysql/libexec/mysqld:
Shutdown complete
```

5 Instalacja standardowa MySQL Cluster

5.1 Dokumentacja kompilacji i instalacji paczki MySQL Cluster

W fazie 2 projektu przygotowana została paczka z instalacją silnika bazy danych zoptymalizowaną pod klaster – MySQL Cluster. Niniejsza dokumentacja techniczna zawiera metody instalacji i konfiguracji tej paczki – paczki nr 2 opisanej w celach projektu oraz przeprowadzone testy pokazujące poprawność działania DBMS.

5.1.1 Kompilacja zródeł

Analogicznie jak w [FN06] do poprawnego działania paczki potrzebna jest kompilacja źródeł z odpowiednimi parametrami, a nastepnie modyfikacja plików konfiguracyjnych. Kod źródłowy przygotowany jest za pomocą `autoconf`, wymagana więc była standardowa procedura:

```
$ configure
$ make
$ make install
```

Podsumowując, do konfiguracji użyto następujących opcji:

```
--prefix=$HOME/mysql
--exec-prefix=$HOME/mysql
--with-unix-socket-path=$HOME/mysql/thesockets
--with-innodb
--with-ndbcluster
--with-ndb-test
--with-ndb-docs
--with-ndb-ccflags=CFLAGS
```

5.1.2 Zmiany ścieżek dostępu

Ponieważ kompilacja odbyła sie na koncie użytkownika `kfabjans`, analogicznie jak w [FN06] należało zamienić w powstałej z kompilacji instalacji ciagi: `/home/rso/kfabjans/` na `$HOME`, tak aby instalując paczkę u dowolnego użytkownika katalog domowy był poprawny.

5.1.3 Przygotowanie skryptów startujących i zamykających usługi

Następnie stworzono skrypty, które będą startować i zamykać usługi:

- **mysql_mgm_server** – uruchamianie serwera zarządzającego **ndb_mgmd**,
- **mysql_ndbd_server** – uruchamianie węzłów danych **ndb**,
- **mysql_server** – uruchamianie serwerów MySQL **mysqld**,
- **mysql_client** – uruchamianie klientów MySQL **mysql**,
- **mysql_mgm_client** – uruchamianie klienta zarządzającego **ndb_mgm**.

Skrypt **mysql_mgm_server** wygląda następująco:

```
#!/bin/sh

export LD_LIBRARY_PATH=$HOME/mysql/lib/mysql
COMMAND="$HOME/mysql/libexec/ndb_mgmd"
LOG="$HOME/mysql/the3.pid"
if test -s $HOME/mysql/the3.pid
then
    MY=1;
    PID=$(cat $HOME/mysql/the3.pid);
else
    MY=0;
fi

case $# in
1)
    if test $1 = "start"
    then
if test $MY -eq 1
then
    echo 'MySQL process is already running...'
    exit 1
fi

if test $MY -eq 0
        then
    ${COMMAND} &
    echo $! >> "$LOG"
    exit 1;
fi
    fi

    if test $1 = "stop"
    then
if test $MY -eq 1
then
    #kill $PID
    killall -9 ndb_mgmd
    rm $HOME/mysql/the3.pid;
    echo 'MySQL process stopped properly...'
    exit 1
else
    echo 'MySQL process already stopped'
    exit 1
```

```
fi
    fi

    if test $1 = "restart"
    then
if test $MY -eq 1
then
    killall -9 ndb_mgmd
    #kill $PID
    rm $HOME/mysql/the3.pid;
    ${COMMAND} &
    echo $! >> "$LOG"
    exit 1
else
    ${COMMAND} &
    echo $! >> "$LOG"
    exit 1
fi
    fi

    echo 'Wrong option, try one of the following ways'
    echo './mysql_mgm_server start'
    echo './mysql_mgm_server stop'
    echo './mysql_mgm_server restart'
    exit 1
    ;;
*)
    echo 'Improper number of arguments, try one of the
following ways'
    echo './mysql_mgm_server start'
    echo './mysql_mgm_server stop'
    echo './mysql_mgm_server restart'
    exit 1
    ;;
esac
```

Skrypt **mysql_ndbd_server** wygląda następująco:

```
#!/bin/sh

export LD_LIBRARY_PATH=$HOME/mysql/lib/mysql
COMMAND="$HOME/mysql/libexec/ndbd"
COMMAND1="$HOME/mysql/libexec/ndbd --initial"
LOG="$HOME/mysql/the2.pid"
if test -s $HOME/mysql/the2.pid
then
```

```
    MY=1;
    PID=$(cat $HOME/mysql/the2.pid);
else
    MY=0;
fi

case $# in
1)
    if test $1 = "start"
    then
if test $MY -eq 1
then
    echo 'MySQL process is already running...'
    exit 1
fi

if test $MY -eq 0
        then
if test -s $HOME/mysql/configs/ndbdinit
then
    ${COMMAND} &
    echo $! >> "$LOG"
    exit 1
else
    ${COMMAND1} &
    echo $! >> "$LOG"
    echo "already initiated" > $HOME/mysql/configs/ndbdinit
    exit
fi
fi
    fi

    if test $1 = "stop"
    then
if test $MY -eq 1
then
    #kill $PID
    killall -9 ndbd
    rm $HOME/mysql/the2.pid;
    echo 'MySQL process stopped properly...'
    exit 1
else
    echo 'MySQL process already stopped'
    exit 1
fi
```

```
    fi

    if test $1 = "restart"
    then
if test $MY -eq 1
then
    #kill $PID
    killall -9 ndbd
    rm $HOME/mysql/the2.pid;
    ${COMMAND} &
    echo $! >> "$LOG"
    exit 1
else
    echo 'MySQL process already stopped'
    ${COMMAND} &
    echo $! >> "$LOG"
    exit 1
fi
    fi

    echo 'Wrong option, try one of the following ways'
    echo './mysql_ndbd_server start'
    echo './mysql_ndbd_server stop'
    echo './mysql_ndbd_server restart'
    exit 1
    ;;
*)
    echo 'Improper number of arguments, try one of the
following ways'
    echo './mysql_ndbd_server start'
    echo './mysql_ndbd_server stop'
    echo './mysql_ndbd_server restart'
    exit 1
    ;;
esac
```

Skrypt **mysql_server** wygląda następująco:

```
#!/bin/sh

if test -s $HOME/mysql/the1.pid
then
    MY=1;
    PID=$(cat $HOME/mysql/the1.pid);
else
    MY=0;
```

```
fi

case $# in
1)
    if test $1 = "start"
    then
if test $MY -eq 1
then
    echo 'MySQL process is already running...'
    exit 1
fi

if test $MY -eq 0
        then
    $HOME/mysql/libexec/mysqld &
    exit 1
fi
    fi

    if test $1 = "stop"
    then
if test $MY -eq 1
then
    kill $PID
    echo 'MySQL process stopped properly...'
    exit 1
else
    echo 'MySQL process already stopped'
    exit 1
fi
    fi

    if test $1 = "restart"
    then
if test $MY -eq 1
then
    kill $PID
    $HOME/mysql/libexec/mysqld &
    exit 1
else
    echo 'MySQL process already stopped'
    $HOME/mysql/libexec/mysqld &
    exit 1
fi
    fi
```

```
    echo 'Wrong option, try one of the following ways'
    echo './mysql_server start'
    echo './mysql_server stop'
    echo './mysql_server restart'
    exit 1
    ;;
*)
    echo 'Improper number of arguments, try one of the
following ways'
    echo './mysql_server start'
    echo './mysql_server stop'
    echo './mysql_server restart'
    exit 1
    ;;
esac
```

Skrypt **mysql_client** wygląda następująco:

```
#!/bin/sh

export LD_LIBRARY_PATH=$HOME/mysql/lib/mysql
$HOME/mysql/bin/mysql $*;
```

Skrypt **mysql_mgm_client** wygląda następująco:

```
#!/bin/sh

export LD_LIBRARY_PATH=$HOME/mysql/lib/mysql
$HOME/mysql/bin/ndb_mgm $*;
```

5.1.4 Główna część paczki

Utworzono archiwum `tar` ze skompilowanej instalacji oraz powyższych stworzonych skryptów i spakowano go, tworząc podstawową część paczki:

```
mysql_install.tar.gz
```

5.1.5 Przygotowanie skryptu instalacyjnego

Przygotowano skrypt instalacyjny, rozpakowujący i konfigurujący paczkę na koncie użytkownika. Instalacja bedzie odbywała się w następujący sposób:

```
#!/bin/sh

if test $# -ne 0
then
```

```
    echo "Improper number of arguments";
    echo "Try install.sh without any arguments";
    exit 1
fi

if test -d $HOME/mysql
then
    echo "MySQL already installed...";
    exit 1
fi

PWD=$(pwd);
#-----------------------------------------------------
```

Następuje rozpakowanie paczki:

```
#-----------------------------------------------------
#extraction of tar content
if test $PWD = $HOME
then
    tar -xvvzf mysql.tar.gz
else
    tar -xvvzf mysql.tar.gz
    cp -R $PWD/mysql $HOME/mysql
    rm -R $PWD/mysql
fi

if test ! -d $HOME/mysql
then
    echo "Corrupted tar archive...";
    exit 1
fi
#-----------------------------------------------------
```

Konieczna jest modyfikacja praw dostępu dla głównego katalogu MySQL:

```
#-----------------------------------------------------
chgrp -R rso1 $HOME/mysql
chmod -R u+wrx $HOME/mysql
chmod -R g+wrx $HOME/mysql
#-----------------------------------------------------
```

Następnie tworzone są pliki konfiguracyjne [con]:

- **config.ini** - konfiguracja serwera zarządzającego i węzłów danych,
- **.my.cnf** - konfiguracja klienta, węzłów danych i scieżka do pliku konfiguracyjnego serwera zarządzającego,

oba w katalogu użytkownika – `$HOME`.

Pliki konfiguracyjne mają prostą strukturę. Krzyżyk (#) oznacza komentarz. Opcje w pliku rozdzielane są nagłówkami, które zawarte są w nawiasach kwadratowych.

Konieczne są również modyfikacje praw dostępu dla plików konfiguracyjnych.

```
#-----------------------------------------------------
#creating config file config.ini

echo "[NDBD DEFAULT]
NoOfReplicas= 1
DataDir= /mnt/disk/rso/rso1/mysql

##########################
#MYSQL-MENAGEMENT-SERVER#
##########################
[NDB_MGMD]
PortNumber=2121
Hostname= lab4-cs2
#DataDir= /mnt/disk/rso/rso1/mysql

####################
#MYSQL-DATA-NODES #
####################
[NDBD]
HostName= lab4-cs2
#DataDir= /mnt/disk/rso/rso1/mysql #tak powinno byc
# a ponizsze ustawienie wymuszone jest obecnymi ustawieniami
# w klastrze gdyz nazwy podmontowanych dyskow nie sa zgodne
DataDir = $HOME/mysql/lib/mysql

#############
#MYSQL_NODES#
#############
[MYSQLD]
[MYSQLD]" > $HOME/config.ini

if test ! -s $HOME/config.ini
then
    echo "Configuration file config.ini cannot be created..";
    exit 1
fi

chgrp rso1 $HOME/config.ini
```

```
chmod u+wrx $HOME/config.ini
chmod g+wrx $HOME/config.ini
#-------------------------------------------------
#creating config file .my.cnf

echo "[client]
###############
#MYSQL-CLIENT#
###############
socket          = $HOME/mysql/thesocket
port            = 2012

[mysqld]
###############
#MYSQL-NODE  #
###############
basedir         = $HOME/mysql
datadir         = $HOME/mysql/lib/mysql
pid-file        = $HOME/mysql/the1.pid
socket          = $HOME/mysql/thesocket
port = 2012
ndbcluster
ndb-connectstring=nodeid=1,host=lab4-cs2:2121

[ndbd]
##################
#MYSQL-DATA_NODE#
##################
connect-string = nodeid=1,host=lab4-cs2:2121

#########################
#MYSQL_MENAGEMENT-SERVER#
#########################
[ndb_mgmd]
config-file = $HOME/config.ini" > $HOME/.my.cnf

if test ! -s $HOME/.my.cnf
then
    echo "Configuration file .my.cnf cannot be created..";
    exit 1
fi

chgrp rso1 $HOME/.my.cnf
chmod u+wrx $HOME/.my.cnf
chmod g+wrx $HOME/.my.cnf
```

```
#-------------------------------------------------
```

Następuje wywołanie właściwej istalacji:

```
#-------------------------------------------------
$HOME/mysql/bin/mysql_install_db
#-------------------------------------------------
```

oraz tworzenie linków i kolejne zmiany praw dostępu:

```
#-------------------------------------------------
#creating symlinks and mysqld deamon service
#ordinary client
ln -s $HOME/mysql/mysql_client $HOME/mysql_client

if test ! -s $HOME/mysql_client
then
    echo "MySQL client is not available";
    exit 1
fi

chgrp rso1 $HOME/mysql_client
chmod u+wrx $HOME/mysql_client
chmod g+wrx $HOME/mysql_client

#server-node
ln -s $HOME/mysql/mysql_server $HOME/mysql_server

if test ! -s $HOME/mysql_server
then
    echo "MySQL server is not available";
    exit 1
fi

chgrp rso1 $HOME/mysql_server
chmod u+wrx $HOME/mysql_server
chmod g+wrx $HOME/mysql_server

#menagement server
ln -s $HOME/mysql/mysql_mgm_server $HOME/mysql_mgm_server

if test ! -s $HOME/mysql_mgm_server
then
    echo "MySQL menagement server is not available";
    exit 1
fi
```

```
chgrp rso1 $HOME/mysql_mgm_server
chmod u+wrx $HOME/mysql_mgm_server
chmod g+wrx $HOME/mysql_mgm_server

#data-node
ln -s $HOME/mysql/mysql_ndbd_server $HOME/mysql_ndbd_server

if test ! -s $HOME/mysql_ndbd_server
then
    echo "MySQL server is not available";
    exit 1
fi

chgrp rso1 $HOME/mysql_ndbd_server
chmod u+wrx $HOME/mysql_ndbd_server
chmod g+wrx $HOME/mysql_ndbd_server

#menagement client

ln -s $HOME/mysql/mysql_mgm_client $HOME/mysql_mgm_client

if test ! -s $HOME/mysql_mgm_client
then
    echo "MySQL server is not available";
    exit 1
fi

chgrp rso1 $HOME/mysql_mgm_client
chmod u+wrx $HOME/mysql_mgm_client
chmod g+wrx $HOME/mysql_mgm_client

echo "";
echo "Installation complete...";
echo "Remeber to edit configuration files .my.cnf and config.ini!";
```

5.1.6 Przygotowanie skryptu dezinstalacyjnego

Przygotowano również skrypt dezinstalacyjny. Dezinstalacja będzie odbywała się w następujący sposób:

```
#!/bin/sh

#removing all files
if test ! -s $HOME/.my.cnf
```

```
then
    echo ".my.cnf was not removed becouse it's not exist";
fi

if test -s $HOME/.my.cnf
then
    rm $HOME/.my.cnf;
    echo "File .my.cnf was removed...";
fi

if test ! -s $HOME/config.ini
then
    echo "config.ini was not removed becouse it's not exist";
fi

if test -s $HOME/config.ini
then
    rm $HOME/config.ini;
    echo "File config.ini was removed...";
fi

if test ! -s $HOME/mysql_client
then
    echo "mysql_client was not removed becouse it's not exist"
fi

if test -s $HOME/mysql_client
then
    rm $HOME/mysql_client
    echo "File mysql_client was removed...";
fi

if test ! -s $HOME/mysql_mgm_client
then
    echo "mysql_mgm_client was not removed becouse it's not exist"
fi

if test -s $HOME/mysql_mgm_client
then
    rm $HOME/mysql_mgm_client
    echo "File mysql_mgm_client was removed...";
fi

if test ! -s $HOME/mysql_server
then
```

```
    echo "mysql_server was not removed becouse it's not exist"
fi

if test -s $HOME/mysql_server
then
    rm $HOME/mysql_server;
    echo "File mysql_server was removed...";
fi

if test ! -s $HOME/mysql_mgm_server
then
    echo "mysql_mgm_server was not removed becouse it's not exist"
fi

if test -s $HOME/mysql_mgm_server
then
    rm $HOME/mysql_mgm_server;
    echo "File mysql_mgm_server was removed...";
fi

if test ! -s $HOME/mysql_ndbd_server
then
    echo "mysql_ndbd_server was not removed becouse it's not exist"
fi

if test -s $HOME/mysql_ndbd_server
then
    rm $HOME/mysql_ndbd_server;
    echo "File mysql_ndbd_server was removed...";
fi

#removing direcotry

if test ! -d $HOME/mysql
then
    echo "mysql directory was not removed becouse it's not exist"
fi

if test -d $HOME/mysql
then
    rm -fR $HOME/mysql;
    echo "Directory mysql was removed";
fi
```

5.2 Konfiguracja MySQL Cluster w warunkach laboratoryjnych

5.3 Instalacja MySQL Cluster

Gotowa paczka jest spakowana i nazywa się:

```
mysql_install.tar.gz
```

Zawiera ona trzy główne pliki:

```
install.sh
uninstall.sh
mysql.tar.gz
```

Pliki należy rozpakować do dowolnego wspólnego katalogu, a następnie uruchomić skrypt `./install.sh`. MySQL Cluster docelowo instaluje się w `$HOME/mysql`.

5.4 Uruchamianie usług

Po instalacji, usługi serwera MySQL należy uruchomić w następujący sposób:

```
$ ./<nazwa_skryptu> start
```

Restart wykonywany jest wydaniem komendy:

```
$ ./<nazwa_skryptu> restart
```

Zatrzymanie następuje po wydaniu komendy:

```
$ ./<nazwa_skryptu> stop.
```

5.4.1 Pliki konfiguracyjne

Opisywane już pliki `config.ini .my.cnf` generuje się w trakcie instalacji. W plikach tych przechowywane są informacje odnośnie konfiguracji MySQL Cluster.

5.4.2 Testy działania

Uruchomiono przygotowane skrypty na koncie użytkownika `jnowacki` i przeprowadzono proste testy, pokazujące działanie zainstalowanego MySQL Cluster [qui]:

```
jnowacki@lab4-cs2:~$ ./mysql_mgm_server start
jnowacki@lab4-cs2:~$ ./mysql_ndbd_server start
jnowacki@lab4-cs2:~$ ./mysql_server start
jnowacki@lab4-cs2:~$ ./mysql_client
jnowacki@lab4-cs2:~$ ./mysql_mgm_client
```

Poniżej przedstawiono efekt przeprowadzonych testów, wskazujące na prawidłową instalację MySQL Cluster.

```
########################################################################
Installing all prepared tables
Fill help tables

To start mysqld at boot time you have to copy
support-files/mysql.server to the right place for your system

PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !
To do so, start the server, then issue the following commands:
/home/rso/kfabjans/mysql/bin/mysqladmin -u root
password 'new-password'
/home/rso/kfabjans/mysql/bin/mysqladmin -u root -h
lab4-cs1 password 'new-password'
See the manual for more instructions.

You can start the MySQL daemon with:
cd /home/rso/kfabjans/mysql ;
/home/rso/kfabjans/mysql/bin/mysqld_safe &

You can test the MySQL daemon with the benchmarks in the
'sql-bench' directory:
cd sql-bench ; perl run-all-tests

Please report any problems with the
/home/rso/kfabjans/mysql/bin/mysqlbug script!

The latest information about MySQL is available on the web at
http://www.mysql.com Support MySQL by buying
support/licenses at https://order.mysql.com

Installation complete...
Remeber to edit configuration files .my.cnf and config.ini!
kfabjans@lab4-cs1:~/paczka$ cd ..
kfabjans@lab4-cs1:~$ ./mysql_mgm_server start
kfabjans@lab4-cs1:~$ ./mysql_ndbd_server start
kfabjans@lab4-cs1:~$ ./mysql_server start
kfabjans@lab4-cs1:~$ 060515 20:38:19
[Note] Starting MySQL Cluster Binlog Thread
InnoDB: The first specified data file ./ibdata1 did not exist:
InnoDB: a new database to be created!
060515 20:38:19  InnoDB: Setting file ./ibdata1 size to 10 MB
InnoDB: Database physically writes the file full: wait...
060515 20:38:19  InnoDB: Log file ./ib_logfile0 did not exist:
new to be created
InnoDB: Setting log file ./ib_logfile0 size to 5 MB
```

```
InnoDB: Database physically writes the file full: wait...
060515 20:38:19  InnoDB: Log file ./ib_logfile1 did not exist:
new to be created
InnoDB: Setting log file ./ib_logfile1 size to 5 MB
InnoDB: Database physically writes the file full: wait...
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: Creating foreign key constraint system tables
InnoDB: Foreign key constraint system tables created
060515 20:38:20  InnoDB: Started; log sequence number 0 0
060515 20:38:20 [Note] /home/rso/kfabjans/mysql/libexec/mysqld:
ready for connections.
Version: '5.1.7-beta'  socket:
'/home/rso/kfabjans/mysql/thesocket'
port: 2012  Source distribution

kfabjans@lab4-cs1:~$ ./mysql_mgm_client
 --connect-string=lab4-cs1:2121
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: lab4-cs1:2121
Cluster Configuration
---------------------
[ndbd(NDB)]     1 node(s)
id=2    @10.0.0.1  (Version: 5.1.7, Nodegroup: 0, Master)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @10.0.0.1  (Version: 5.1.7)

[mysqld(API)]   2 node(s)
id=3    @10.0.0.1  (Version: 5.1.7)
id=4 (not connected, accepting connect from any host)

ndb_mgm> exit
kfabjans@lab4-cs1:~$ ./mysql_client
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 5.1.7-beta

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
```

```
| test               |
+--------------------+
2 rows in set (0,00 sec)

mysql> exit
Bye
kfabjans@lab4-cs1:~$ ./mysql_client -u root Welcome to
the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 5.1.7-beta

Type 'help;' or '\h' for help. Type '\c' to clear the
buffer.

mysql> create database test2
    -> ;
Query OK, 1 row affected (0,18 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| cluster_replication |
| mysql              |
| test               |
| test2              |
+--------------------+
5 rows in set (0,01 sec)

mysql> exit
Bye
kfabjans@lab4-cs1:~$
```

5.5 Wyniki testów

W ramach realizacji fazy II przeprowadzona została analiza dostępnych metod i narzędzi testujących bazy danych MySQL. Efekty rozpoznania przedstawione zostały w dalszej części dokumentacji. Dokonano wyboru narzędzia testującego, dokonano autorskich zmian według własnych scenariuszy testów. Właściwe testy wydajnościowe oraz naciskowe przeprowadzone zostaną w ramach prac nad docelowym rozwiązaniem autorskim i przedstawione w sprawozdaniu III fazy projektu.

6 Instalacja MySQL Cluster SSI

6.1 Dokumentacja kompilacji i instalacji paczki MySQL Cluster SSI

W fazie 3 projektu przygotowana została paczka z silnikiem bazy danych skonfigurowanym i rozszerzonym o nowy kod (wykorzystujący środowisko OpenSSI). Niniejsza dokumentacja techniczna zawiera metody instalacji i konfiguracji tej paczki – paczki nr 3 opisanej w celach projektu.

6.1.1 Źródła

Podstawą paczki 3 jest kompilacja MySQL Cluster opisana we wcześniejszej częsci dokumentu, wykonana z opcjami:

```
--prefix=$HOME/mysql
--exec-prefix=$HOME/mysql
--with-unix-socket-path=$HOME/mysql/thesockets
--with-innodb
--with-ndbcluster
--with-ndb-test
--with-ndb-docs
--with-ndb-ccflags=CFLAGS
```

6.1.2 Demon i jego konfiguracja

W ramach etapu 3 stworzono demona `ssimysql` oraz plik konfiguracyjny `ssimysql.cnf`, z którego będzie korzystał. W pliku konfiguracyjnym znajdują się dane o ilości serwerów zarządzających, ilości replik oraz ilości węzłów. W etapie drugim prac nad projektem stworzono skrypty startujące i zamykające usługi:

- **mysql_mgm_server** – serwer zarządzający **ndb_mgmd**,
- **mysql_ndbd_server** – węzły danych **ndb**,
- **mysql_server** – serwery MySQL **mysqld**,
- **mysql_client** – klienty MySQL **mysql**,
- **mysql_mgm_client** – klient zarządzający **ndb_mgm**.

Nie uległy one zmianie i są wykorzystywane przez stworzonego demona. Demon wykonuje `node balance` i uruchamia w podanej kolejności na najmniej obciążonych węzłach:

- serwery zarządzające
- repliki
- zwykłe węzły

Po uruchomieniu usług demon monitoruje stan odpalonych procesów porównując go z zadanym plikiem konfiguracyjnym. Jeśli cos nie zgadza to uruchamiana jest brakująca usługa na najmniej obciążonym węźle.

6.2 Instalacja MySQL Cluster SSI

Gotowa paczka jest spakowana i nazywa się:

```
mysql_install.tar.gz
```

Analogicznie jak w przypadku instalacji standardowego rozwiązania MySQL Cluster, pliki należy rozpakować do dowolnego wspólnego katalogu, a następnie uruchomić skrypt `./install.sh`.

6.3 Uruchamianie demona

Po instalacji, nie należy juz uruchamiać kolejno usług, tak jak we wczesniej opisanej instalacji MySQL Cluster. Należy natomiast uruchomić demona w taki sposób jak uruchamiało się usługi:

```
$ ./ssimysql start
```

Restart wykonywany jest wydaniem komendy:

```
$ ./ssimysql restart
```

Zatrzymanie następuje po wydaniu komendy:

```
$ ./ssimysql stop.
```

7 Koncepcja autorskiego rozwiązania klastrowego

Sekcja ta omawia koncepcję autorskiego rozwiązania klastrowego realizowanego w ramach projektu. Opisane zostały założenia przyjęte na samym początku opracowywania koncepcji. Przeanalizowano stan obecny w zakresie istniejących rozszerzeń klastrowym dla MySQL. Zdefiniowano cele, jakie powinno spełniać projektowane rozwiązanie, omówiono szczegółowo koncepcję i ustalono przewidywane korzyści wynikające z implementacji, w celu późniejszej weryfikacji.

7.1 Założenia wstępne

Czynności polegające na zainstalowaniu systemu MySQL Cluster w środowisku OpenSSI, analizie dokumentacji oraz obserwacji działania systemu w praktycznym zastosowaniu pozwoliła stworzyć wyobrażenie funkcjonowania klastrowego rozwiązania bazodanowego. Również zapoznanie się z działaniem i specyficzną funkcjonalnością środowiska OpenSSI przyczyniło się do wyodrębnienia jego cech ważnych z punktu widzenia stworzenia systemu, którego przeznaczeniem będzie działanie jak najbardziej zbliżone do optymalnego.

Należy postawić jasno szereg pytań, na jakie zespół projektujący rozwiązanie będące przedmiotem projektu musi znaleźć odpowiedź, by wybrać optymalną ścieżkę prac projektowych. Poniżej przedstawiono zbiór zagadnień, które wzięto pod uwagę przed przystąpieniem do planowania koncepcji autorskiego rozwiązania klastrowego systemu bazodanowego.

1. Jak wygląda obecny stan wiedzy w dziedzinie klastrowych systemów bazodanowych (opartych o rozwiązania MySQL) dedykowanych dla środowiska OpenSSI?
2. Czy oferowane cechy rozwiązania MySQL Cluster mogą stanowić bazę dla opracowania docelowej koncepcji autorskiej?
3. Czy możliwe jest stworzenie od podstaw rozwiązania dedykowanego dla środowiska OpenSSI o cechach wyraźnie lepszych niż MySQL Cluster?
4. Jaki jest status prac nad ciągle rozwijanym rozwiązaniem MySQL Cluster?
5. Jak wykorzystać zalety środowiska OpenSSI do stworzenia optymalnego rozwiązania docelowego?
6. Jakie powinny być silne strony docelowego systemu bazodanowego w odniesieniu do systemów już istniejących?

Udzielenie odpowiedzi na powyższe pytania, poparte zbieraniem wiedzy w dostępnych książkach, opracowaniach oraz sieci Internet, pozwoliło odpowiedzieć na nadrzędne pytanie: *jak powinno wyglądać docelowe rozwiązanie klastrowego systemu bazodanowego?*

Według naszej wiedzy nie istnieją systemy bazodanowe zoptymalizowane do pracy w środowisku OpenSSI, oparte na rozwiązaniu MySQL. Jedynym powiązaniem systemu MySQL Cluster ze środowiskiem OpenSSI, jakie udało się odnaleźć w sieci Internet, był opis konfiguracji wcześniejszych wersji MySQL Cluster do pracy w środowisku OpenSSI. Brakuje też informacji o innych

rozwiązaniach bazodanowych, o których można by powiedzieć, że stworzone zostały do pracy w tym specyficznym środowisku klastrowym. Pozwala to postawić tezę, że projekt realizowany jest w warunkach pionierskich, a wszelkie zakończone sukcesem prace mające na celu wykorzystanie cech OpenSSI do stworzenia optymalnego klastrowego systemu bazodanowego opartego na rozwiązaniach MySQL wniesie nową wiedzę i doświadczenie do tej dziedziny informatyki. W tych szczególnych warunkach niebezpieczne jest decydowanie się na odważne koncepcje, rewolucyjne w odniesieniu do istniejących rozwiązań. Takie podejście mogłoby zakończyć się niepowodzeniem i nie wnieść do tematu pożytecznych informacji. Natomiast koncepcja oparta na realnych założeniach, mających szansę implementacji w warunkach, w jakich pracuje zespół projektowy, może przynieść sukces, który stanowił będzie podstawę do dalszych prac w tej dziedzinie.

MySQL AB dostarcza rozwiązanie MySQL Cluster, przeznaczone dla środowisk klastrowych. Jednak architektura tego rozwiązania traktuje klaster jako zbiór komputerów połączonych w sieć, nie uwzględniając znacznie dalej idących cech, jakimi charakteryzuje się środowisko OpenSSI. Rozwiązanie to jest również w fazie rozwoju, co oznacza, że część jego funkcjonalności nie jest jeszcze doskonała, lub też w ogóle nie istnieje w tym momencie jej implementacja, mimo iż jest przewidziana w ogólnej koncepcji. Pojawia się więc wątpliwość, czy sensowne jest opieranie się na takim rozwiązaniu, czy też warto pokusić się o stworzenie koncepcji od podstaw w oparciu o dopracowane mechanizmy systemu bazodanowego MySQL bez rozszerzeń klastrowych. Biorąc pod uwagę, że rozwiązania instytucji jaką jest MySQL AB cieszą się uznaniem i zaufaniem różnych środowisk, przyjęcie koncepcji tworzenia autorskiego rozwiązania od podstaw wydaje się nierozsądne. Rozwiązanie takie musiałoby przejść szereg testów udowadniających jego poprawność i wyższość w odniesieniu do rozwiązań już istniejących. Co więcej, rozwiązanie takie wymaga ogromnej wiedzy na temat mechanizmów systemu MySQL. Wreszcie tworzenie koncepcji od podstaw mogłoby doprowadzić do rozwiązania skrajnie odmiennego od tego proponowanego w ramach MySQL Cluster, a nie ma podstaw by twierdzić, że łatwo jest stworzyć system pod wieloma względami lepszy niż ten będący efektem pracy osób najlepiej zorientowanych w mechanizmach silnika bazodanowego MySQL. Alternatywnie mogłoby się okazać, że powstałaby koncepcja bardzo zbliżona do istniejących, cały wysiłek okazałby się więc próżny i nic nie wnoszący do dziedziny zainteresowania. Uznano więc, że zasadne jest przyjęcie koncepcji polegających na rozwijaniu już istniejącego rozwiązania MySQL Cluster, tym bardziej, że posiada ono cechy sprawdzające się w środowisku klastrowym, jako szczególnym przypadku środowiska sieciowego, do jakiego zostało zaprojektowane.

Decydując się na rozbudowę rozwiązania znajdującego się w fazie rozwoju należy śledzić zakres prac nad udoskonalaniem tego rozwiązania. Niepożądaną sytuacją jest powielanie prac, gdyż prowadzi do stworzenia rozwiązań niezgodnych ze sobą. W tworzeniu koncepcji rozwiązania klastrowego należy więc uwzględnić kontekst prac nad systemem MySQL Cluster, opisany w jego dokumentacji technicznej. Autorzy koncepcji świadomi trwających prac podjęli stosowne decyzje o konkretnych zakresach koncepcji docelowego rozwiązania autorskiego.

Bezcelowe jest projektowanie nowego systemu bez założenia, w czym ma on być lepszy od rozwiązań istniejących. Konieczne jest zdefiniowanie celów i wymagań stawianych przed tworzonym systemem. W przypadku systemu bazodanowego możemy mówić o dwóch ważnych aspektach jego działania, które podlegać mogą ulepszaniu. Pierwszym z nich jest wydajność,

drugim zaś niezawodność. W kontekście wydajności możemy uzyskać większą efektywność wykonywania zapytań, krótsze czasy odpowiedzi przy różnym obciążeniu, możliwość obsługi większej liczby klientów jednocześnie. Przez poprawianie niezawodności systemu rozumie się zwiększoną odporność na awarię, wysoką dostępność, itp. Wykorzystanie mocy obliczniowej wielu komputerów znajdujących się w klastrze samo w sobie powoduje zwiększenie wydajności w stosunku do rozwiązania opartego na pojedynczym serwerze. Autorzy założyli więc, że warto skupić się na kwestii niezawodności klastrowego systemu bazodanowego, zwiększanie wydajności traktując jako kwestię drugorzędną. Od docelowego rozwiązania oczekuje się więc, że będzie bardziej odporne na awarie niż istniejące rozwiązania oferując wysoką dostępność w warunkach celowych i losowych uszkodzeń.

Podsumowując, opracowując koncepcję autorskiego rozwiązania klastrowego systemu bazodanowego opartego o silnik MySQL, zespół przyjął następujące założenia:

1. Punktem wyjścia dla docelowego rozwiązania będzie istniejący system MySQL Cluster.
2. W trakcie prac projektowych brany będzie kontekst rozwoju rozwiązania MySQL Cluster, w celu uniknięcia tworzenia rozwiązania, które mogłoby stać się niezgodne z systemem podstawowym.
3. W opracowaniu koncepcji rozwiązania docelowego położony zostanie nacisk na wykorzystanie cech istniejącego systemu MySQL Cluster i dokonanie zmian pozwalających czerpać korzyści ze specyficznej funkcjonalności środowiska OpenSSI.
4. Wzrost wartości rozwiązania docelowego w stosunku do rozwiązań istniejących polegał będzie na zwiększeniu niezawodności systemu, przy czym zwiększanie jego wydajności jest kwestią drugorzędną.

W oparciu o powyższe założenia zidentyfikowany został stan obecny, a więc cechy istniejącego rozwiązania i opracowana została koncepcja autorskiego rozwiązania docelowego.

7.2 Punkt wyjściowy - standardowa instalacja MySQL Cluster

Zgodnie z opisanymi założeniami, punktem wyjściowym docelowego rozwiązania autorskiego jest standardowa istalacja MySQL Cluster z mechanizmem składowania danych NDB. W skład tego rozwiązania wchodzą następujące składniki:

- węzeł zarządzający (ang. *management node*) - konfiguracja i zarządzanie klastrem,
- węzły danych (ang. *data nodes*) - realizacja mechanizmu składowania danych NDB - przechowywanie danych,
- węzły SQL (ang. *SQL nodes*) - obsługa zapytań SQL.

Według twórców, MySQL Cluster charakteryzuje się wysoką dostępnością (na poziomie 99,999%). Z punktu widzenia docelowego rozwiązania istotne są następujące jego cechy:

- **redundancja danych** - rozwiązanie przewiduje tworzenie wielu pełnych replik tych samych danych lub replik w ramach grup węzłów, z których każda przechowuje tylko część (partycję) danych. Redundancję realizuje się poprzez uruchomienie kilku węzłów danych w klastrze, przy odpowiednim uwzględnieniu tego w pliku konfiguracyjnym węzła zarządzającego.
- **synchroniczna replikacja** - najważniejsza z cech rozwiązania zapewniająca pełną spójność danych (przy założeniu działania węzła zarządzającego). Dane w MySQL Cluster replikowane są synchronicznie, tj. każda zmiana danych propagowana jest automatycznie na wszystkie węzły danych. System bazodanowy oparty o to rozwiązanie gwarantuje więc pełną spójność danych.
- **rozproszona architektura bez pojedynczego punktu awarii** - system MySQL Cluster składa się z niezależnych elementów, których zwielokrotnienie zwiększa odporność na awarię. Poszczególne węzły mogą być zatrzymywane, uruchamiane ponownie podczas gdy cały system funkcjonuje nieprzerwanie, świadcząc usługi aplikacjom klienckim.
- **wysoka wydajność** - wysoka wydajność MySQL Cluster zapewniona poprzez przechowywanie danych w pamięci operacyjnej. Zwiększa to wymagania sprzętowe stawiane środowisku klastrowemu, jednak efektem jest eliminacja wąskich gardeł związanych z obsługa operacji IO, skrócenie czasów odpowiedzi i zwiększenie liczby obsługiwanych transakcji w stosunku do innych rozwiązań.

Powyższe cechy zostały uznane za wystarczające, by przyjąć MySQL Cluster za bazę do stworzenia autorskiego rozwiązania klastrowego, co jest celem tego projektu. Część z nich, jak redundancja danych i rozproszona architektura, wykorzystana zostanie w pełni, podczas gdy z wysokiej wydajności częściowo można zrezygnować na rzecz przechowywania danych w przestrzeniach tabel dyskowych.

Projektując rozwiązanie oparte o MySQL Cluster należy być świadomym, iż jest to system w fazie rozwoju. Na obecnym etapie istnieje wiele ograniczeń wyspecyfikowanych przez twórców w dokumentacji. Do najważniejszych zaliczyć należy:

- niezgodności składniowe z innymi serwerami MySQL
  - brak wsparcia indeksów textowych (ang. *text indexes*),
  - brak możliwości użycia kolumny typu BIT jako klucza głównego lub jego części,
  - ograniczenie długości zapytania CREATE TABLE do 4096 znaków (*usunięte począwszy od wersji 5.1.9*),
- wsparcie obsługi transakcji jedynie na poziomie READ COMMITED,
- brak częściowego wycofania transakcji,
- jeśli zapytanie SELECT odnosi się do kolumn typu BLOB, TEXT lub VARCHAR poziom izolacji ogranicza się do odczytu z blokadą, co nie gwarantuje spójności danych,
- maksymalna liczba tabel ograniczona do 1792,

- maksymalna liczba atrybutów tabel ograniczona do 128,
- ograniczenia w długości nazw baz danych, table, atrybutów,
- problemy wydajnościowe spowodowane sekwencyjnym dostępem do mechanizmu NDB,
- brak możliwości zmiany schematu bazy danych on-line,
- maksymalna liczba węzłów danych ograniczona do 48,
- maksymalna liczba wszystkich węzłów (dowolnego typu) w klastrze ograniczona do 63.

Warto też pamiętać o czysto technicznym ograniczeniu czy też raczej wymaganiu. W związku z faktem, iż dane w mechanizmie NDB przechowywane są w pamięci operacyjnej, rozmiar dostępnej pamięci RAM na każdym z węzłów danych w zależności od rozmiaru bazy danych musi być równy co najmniej:

$$(SizeOfDatabase \times NumberOfReplicas \times 1.1) \div NumberOfStorageNodes \qquad (1)$$

Przystępując do opracowywania koncepcji rozwiązania opartego o MySQL Cluster, należy też mieć na uwadze, że system ten zaprojektowany został na klastry, w których każdy z komputerów jest odrębną jednostką o znanym adresie IP. OpenSSI jako rozwiązanie klastrowe idzie krok dalej, oferując ideę *single system image*, dającą spójny obraz jednego systemu, w rzeczywistości rozproszonego na wielu komputerach klastra. Daje to szerego nowych możliwości migracji procesów, komunikacji międzyprocesowej i innych mechanizmów mogących wpłynąć na kształt i jakość ostatecznego rozwiązania bazodanowego.

Analiza działania systemu MySQL Cluster w świetle wyżej opisanego faktu doprowadziła do zauważenia, że rozwiązanie to zostało przeznaczone do działania w statycznej, z góry ustalonej plikami, konfiguracji. W plikach konfiguracyjnych wpisane są adresy IP węzłów danego typu. W przypadku awarii któregoś z nich nie są podejmowane żadne czynności mające przywrócić wcześniejszy stan klastra, nawet gdy istnieją zasoby (dodatkowe, wolne komputery), które by na to pozwalały. Redundancja oczywiście zapobiega awarii całego systemu w przypadku awarii pojedynczego elementu, jednak awaria kilku elementów jednocześnie (na przykład odcięcie segmentu sieci) w niefortunnych okolicznościach może spowodować zaprzestanie działania całego rozwiązania. Co więcej, awaria niektórych węzłów może nie wpływać na całość w kontekście oferowania funkcjonalności usług, jednak może spowodować nieprawidłowe zachowanie systemu, co może mieć jeszcze gorsze konsekwencje niż jawna awaria. Dzieje się tak na przykład w przypadku awarii komputera na którym działa węzeł zarządzających. Gdy węzeł ten nie działa system nadal oferuje pełną funkcjonalność, jednak nie zachodzi prawidłowa replikacja danych. W efekcie otrzymujemy efekt tzw. *podzielonego mózgu*, tj. dane przechowywane w poszczególnych węzłach danych są całkowicie niespójne.

7.3 Cel autorskiego rozwiązania klastrowego

W oparciu o opisane powyżej założenia i analizę punktu wyjściowego do podjęcia dalszych prac projektowych, autorzy zdefiniowali cele, które zamierzają zrealizować w ramach projektu. Jak

wspomniano wcześniej, rozwiązanie MySQL Cluster oferuje znaczny wzrost wydajności w stosunku do serwera MySQL instalowanego na pojedynczym komputerze. Ma to związek z faktem, że mechanizm NDB działa w pamięci operacyjnej. Testy (*sekcja 2.3*) wskazują na rzeczywisty istotny wzrost wydajności MySQL Cluster w stosunku do serwera MySQL. Brak jest przesłanek, które wskazywałyby, że specyficzne funkcje środowiska OpenSSI w znaczący sposób mogłbyby wpłynąć na poprawę tej wydajności.

Wręcz przeciwnie, wysoka wydajność MySQL Cluster okupiona jest wysokimi wymaganiami dotyczącymi pamięci operacyjnej dostępnej dla węzłów danych. Autorzy zdecydowali, że przynajmniej część danych powinna być przechowywana w przestrzeniach tabel składowanych na dyskach twardych. Począwszy od wersji 5.1.6, MySQL Cluster oferuje mechanizmy składowania nieindeksowanych kolumn tabel NDB na dysku, zamiast w pamięci RAM. Doprowadzenie do takiego stanu jest jednym z celów projektu. Może to oznaczać spadek wydajności docelowego rozwiązania związany z koniecznością wykonywania czasochłonnych operacji IO. Jednak oczekuje się, że wydajność pozostanie na poziomie wyższym niż w przypadku serwera MySQL bez rozszerzeń klastrowych przez wzgląd na rozproszenie aplikacji, redundancję danych i węzłów oraz równoważenie obciązenia.

Drugą kwestią w obszarze zainteresowań zespołu projektowego jest zwiększenie niezawodności projektowanego rozwiązania. Za cel autorzy postawili utrzymanie działania całego systemu jak długo działa przynajmniej jeden komputer klastra. Takie rozwiązanie mogłoby być uznane za wysoko odporne na awarie.

Duży nacisk w projektowaniu koncepcji położono na dynamiczną reakcję rozwiązania na różnego typi anomalne zdarzenia. Docelowo projektowane rozwiązanie powinno sprawnie reagować na zdarzenia takie jak awaria węzła, odłączenie od sieci i ponowne przyłączenie. Byłaby to znacząca przewaga w odniesieniu do obecnego rozwiązania MySQL Cluster, w którym cała konfiguracja jest statyczna.

Ostatnim elementem projektowanego rozwiązania jako systemu rozproszonego jest zachowanie całkowitej przezroczystości dla klienta. Idealnie, klient nie powinien zauważać żadnych różnic korzystając ze standardowego serwera MySQL i rozwiązania opartego na MySQL Cluster, pomijając oczywiście większą wydajność i niezawodność tego drugiego.

Podsumowując, cele jakie powinno spełniać docelowe rozwiązanie autorskie przedstawiają się następująco:

1. Zwiększenie niezawodności poprzez dynamiczne reagowanie na awarie występujące w klastrze.
2. Zachowanie przezroczystości rozwiązania z punktu widzenia klienta MySQL.
3. Zachowanie wysokiej wydajności środowiska MySQL Cluster poprzez redundancję danych i serwisów, równoważenie obciążenia.
4. Zmniejszenie zapotrzebowania rozwiązania na pamięć RAM poprzez składowanie danych na dysku twardym.

W dalszej części dokumentacji opisana została koncepcja rozwiązania, które w zamyśle autorów spełni powyższe postulaty w kontekście opisanych wcześniej założeń.

7.4 Koncepcja autorskiego rozwiązania klastrowego

Koncepcja autorskiego rozwiązania klastrowego opiera się na instalacji standardowego systemu MySQL Cluster z rozszerzeniami w postaci demonów oraz skryptów konfiguracyjnych realizujących, przy wykorzystaniu specyficznych cech i funkcjonalności środowiska OpenSSI, następujące cele projektu:

1. dynamiczne zarządzanie procesami klastrowego systemu bazodanowego,
2. konfiguracja systemu w celu przechowywania danych w tabelach dyskowych,
3. przezroczystość rozproszonego rozwiązania dla klientów MySQL,
4. równoważenie obciążenia usług obsługujacych zapytania SQL.

Cel 1. według prezentowanej koncepcji realizowany jest poprzez implementację demona monitorującego stan procesów, zarządzającego dynamicznie bieżącą konfiguracją klastrowego systemu bazodanowego.

Realizacja celu 2. polega na stworzeniu skryptów konfigurujących MySQL Cluster do przechowywania nieindeksowanych tabel NDB w obszarze trwałej pamięci dyskowej.

Cele 3. oraz 4. mogą być realizowane przy użyciu standardowych rozszerzeń OpenSSI w postaci mechanizmu Cluster Virtual IP Address, umożliwiającego dostepność zdefiniowanych usług klastra pod jednym wspólnym adresem IP przy dodatkowym równoważeniu obciążenia.

7.4.1 Początkowa konfiguracja MySQL Cluster

Na potrzeby projektu założono następującą konfigurację początkową systemu MySQL Cluster:

- 1 węzeł zarządzający (ang. *management node*)
- 3 węzły danych (ang. *data node*)
- 2 węzły SQL (ang. *SQL node*)

Uzyskano w ten sposób zarówno redundancję danych jak i redundancję usług. Każdy z węzłów uruchamiany jest na oddzielnym komputerze, aby spełnić założenie o rozproszeniu systemu. W idealnym rozwiązaniu każdy węzeł danych powinien mieć niezależną partycję do składowania tabel dyskowych. W środowisku laboratoryjnym istnieją 3 niezależne partycje, każdy węzeł danych powinien więc korzystać z innej partycji.

Założeniem projektowanej koncepcji jest dostęp implementowanego demona zarzadzającego do plików konfiguracyjnych węzłów systemu MySQL Cluster. W tym celu skonfigurowano instalację systemu tak by pliki konfiguracyjne węzłów danych i węzłów SQL *.my.cnf* przechowywane były w katalogu domowym użytkownika, natomiast plik konfiguracyjny węzła zarządzającego *configure.ini* w podkatalogu *mysql* w katalogu domowym użytkownika.

7.4.2 Demon zarządzający konfiguracją MySQL Cluster

Celem implementacji demona zarządzającego konfiguracją MySQL Cluster jest reagowanie na sytuacje awaryjne zaistniałe w klastrze. Demon ten monitoruje uruchomione procesy węzłów MySQL Cluster, a w przypadku wykrycia ich awarii uruchamia nowe usługi na dostępnych zasobach klastra, jednocześnie zmieniając w razie potrzeby pliki konfiguracyjne węzłów oraz restartując pozostałe usługi.

Plik konfiguracyjny Demon zarządzający działa w oparciu o plik konfiguracyjny, w którym zawarta jest początkowa konfiguracja MySQL Cluster, a więc adresy IP komputerów, na których uruchomione zostaną początkowo wszystkie węzły systemu. Dodatkową informacją jest minimalna liczba węzłów każdego typu. Plik konfiguracyjny mógłby więc wyglądać następująco:

```
#Initial configuration
MGM    1    10.0.0.20
DN     2    10.0.0.21
DN     3    10.0.0.22
DN     4    10.0.0.23
SQL    5    10.0.0.24
SQL    6    10.0.0.25
#Number of nodes
MGM    1
DN     3
SQL    2
```

Struktura linii pliku konfiguracyjnego w sekcji *#Initial configuration* wygląda więc następująco:

```
typ węzła    ID węzła     adres IP komputera docelowego
```

Struktura linii pliku konfiguracyjnego w sekcji *#Number of nodes* to:

```
typ węzła     minimalna liczba węzłów danego typu
```

Uruchomienie demona Uruchomienie demona następuje poprzez wywołanie z linii poleceń nazwy pliku wykonawczego lub skrytpu. Po uruchomieniu demon wczytuje plik konfiguracyjny i sprawdza poprawność jego struktury. Następnie sprawdza stan klastra pod kątem dostępności zasobów względem wczytanej konfiguracji. Istnieje założenie, że w początkowej konfiguracji każdy węzeł uruchamiany jest na oddzielnym komputerze. Demon zarzadzający powinien być uruchomiony również na innym komputerze. Minimalna liczba dostępnych komputerów w klastrze równa jest więc łącznej liczbie węzłów wynikającej z pliku konfiguracyjnego plus jeden. Konfiguracja powinna też zawierać adresy IP komputerów innych niż ten, z którego uruchamiany jest demon zarządzający. W przypadku spełnienia powyższych warunków, według wskazanej konfiguracji, demon tworzy pliki konfiguracyjne w znanej lokacji i uruchamia usługi MySQL Cluster. W przeciwnym wypadku odmówi uruchomienia usług zwracając przy tym odpowiedni komunikat błędu.

Uwaga! Zgodnie z zaleceniami dokumentacji MySQL węzły uruchamiane są w kolejności:

1. węzeł zarządzający,
2. węzły danych,
3. węzeł SQL.

Uwaga! Po uruchomieniu procesu demon zapamiętuje jego *PID* w celu późniejszego monitorowania.

Usługi MySQL Cluster w środowisku klastrowym Powyżej opisane zostały warunki początkowe dotyczące wymaganej liczby komputerów dostępnych w klastrze. Jednak w trakcie działania w wyniku awarii może się okazać, że liczba komputerów nie będzie wystarczająca. Wolą autorów koncepcji jest by w takiej sytuacji system dalej działał. Konieczne są więc następujące założenia:

1. Każdy węzeł MySQL Cluster powinien być uruchomiony na innym komputerze, jeżeli jest taka możliwość.
2. Jeżeli liczba komputerów dostępnych w klastrze w wyniku awarii jest mniejsza niż liczba węzłów MySQL Cluster możliwe jest uruchomienie więcej niż jednego węzła na jednym komputerze.
3. Na jednym komputerze nie może być uruchomiony więcej niż jeden węzeł tego samego typu.
4. W sytuacji wymagającej uruchomienia więcej niż jednego węzła danego typu na jednym komputerze dodatkowy węzeł nie jest uruchamiany mimo ustalonej miminalnej liczby węzłów danego typu.
5. W brzegowym przypadku dostępności tylko jednego komputera, na komputerze tym uruchamiany jest jeden węzeł danych, jeden węzeł SQL oraz jeden węzeł zarządzający.

Wykrycie awarii Demon zarządzający pamięta *PID* wszystkich uruchomionych węzłów. Dzięki pojedynczej przestrzeni procesów (ang. *single process spaces*) środowiska OpenSSI demon może monitorować stan wszystkich procesów uruchomionych na klastrze. Wykrycie awarii danego węzła polega więc na zauważeniu sytuacji, w której proces węzła przestał działać. Identyfikowany jest typ węzła, który uległ awarii, zmniejszana jest zmienna informująca o liczbie działających węzłów danego typu, której wartość jest następnie porównywana z wartością minimalną, ustaloną w konfiguracji. Gdy wynik porównania oznacza przekroczenie granicy podejmowana jest reakcja na awarię.

Dynamiczna reakcja na awarię W przypadku wykrycia awarii demon podejmuje akcje prowadzące do przywrócenia wyjściowego stanu klastra lub też stanu minimalnego opisanego w konfiguracji, rozumiejąc przez to przywrócenie minimalnej liczby działających węzłów danego typu. Demon porównuje listę dostępnych komputerów klastra z własnymi informacjami o uruchomionych procesach. Na tej podstawie zdobywa wiedzę, na których komputerach może uruchomić proces węzła, który uległ awarii. Wybrany zostaje najmniej obciążony z dostępnych wolnych

komputerów (obciążenie może zostać sprawdzone przy wykorzystaniu polecenia `LOADS` zaimplementowanego w OpenSSI. Zmienione zostają pliki konfiguracyjne, uruchomiony zostaje nowy proces węzła na wybranym komputerze klastra a następnie pozostałe usługi zostają zmuszone do załadowania nowych plików konfiguracyjnych. Odbywa się to poprzez wysłanie sygnału *SIGHUP*. Sygnał *SIGHUP* powoduje przeładowanie pliku konfiguracyjnego procesu bez kończenia działania tego procesu. Dzięki temu obsługiwane połączenia nie zostają przerwane.

UWAGA! Przed uruchomieniem procesu węzła na nowej maszynie sprawdzane jest czy na tej maszynie nie działa już proces danego typu, który nie byłby uwzględniony w obecnej konfiguracji. Istnienie takiego procesu oznaczałoby działanie wcześniej uruchomionej usługi, która została odłączona i ponownie przyłączona. Proces taki jest zabijany przed uruchomieniem nowego.

UWAGA! Nowe procesy zostają uruchomione zgodnie z założeniami zawartymi w akapicie *Usługi MySQL Cluster w środowisku klastrowym*.

UWAGA! Pliki konfiguracyjne są modyfikowane a usługi uruchamiane ponownie według zasad opisanych w akapicie *Zasady ponownego uruchamiania usług.*

Zasady ponownego uruchamiania usług W systemie MySQL Cluster działają 3 typy węzłów. Każdy z nich może ulec awarii, jednak różne są konsekwencje awarii poszczególnych węzłów dla węzłów pozostałych. Poniższe zasady opisują procedurę zmiany plików konfiguracyjnych i ponownego uruchamiania usług danego typu w zależności od typu awarii:

- W przypadku awarii węzła zarządzającego zmieniane są pliki konfiguracyjne węzłów danych i węzłów SQL, po czym następuje przeładowanie plików konfiguracyjnych działających procesów.
- W przypadku awarii węzła SQL zmieniany jest plik konfiguracyjny węzła zarządzającego, który następnie jest przeładowywany. Procesy węzłów danych oraz inne procesy węzłów SQL działają bez zmian.
- W przypadku awarii węzła danych zmieniany jest plik konfiguracyjny węzła zarządzającego, który następnie jest przeładowywany. Procesy innych węzłów danych oraz procesy węzłów SQL działają bez zmian.

Uruchomienie nowego węzła danych Dzięki zastosowaniu sygnału *SIGHUP* przeładowanie pliku konfiguracyjnego węzła danych nie powoduje jego ponownego uruchomienia, co eliminuje konieczność synchronizacji danych. Jednak ponowne uruchomienie procesu węzła danych po awarii wymaga już synchronizacji z działającymi replikami. Obecna wersja MySQL Cluster implementuje algorytm inicjowania węzła danych opisany w [Ron05].

Sytuacja odłączenia i ponownego przyłączenia Sytuacja odłączenia maszyny, na której działa proces któregoś z węzłów a następnie ponownego jej przyłączenia obsłużona jest przez założenie dotyczące sprawdzania istnienia procesów przed uruchomieniem nowych procesów. Jeśli na komputerze, na którym po awarii działa już proces węzła danego typu, jest on zabijany przed

uruchomieniem nowego procesu. Sprawdzane jest również działanie procesów węzłów pozostałych typów, które nie znajdują się w aktualnej konfiguracji. Procesy takie dla porządku również są zabijane.

Wysoka dostępność demona Poprawność opisanej koncepcji jest ściśle uzależniona od działania demona zarządzającego. Jego awaria uniemożliwia dynamiczną rekonfigurację klastra. Nie oznacza to całkowitą awarię systemu, jednak bez demona zarządzającego rozwiązanie sprowadza się do standardowej instalacji MySQL Cluster, które prawdopodobnie nadal będzie długo działało.

Pełna odporność na awarię musi uwzględniać mechanizm zapewnienia działania demona zarządzającego. Środowisko OpenSSI udostępnia mechanizmy zapewniające dostępność aplikacji. Poniżej zamieszczono opis demona realizującego tę funkcjonalność. Opis zaczerpnięty został z [Naj].

Demon *keepalive* monitoruje procesy i demony, które są zarejstrowane przy użyciu interfejsu *spawndaemon*. Kiedy zarejestrowany proces lub demon zawodzi, keepalive zapisuje zdarzenie używając do tego funkcji syslog i wywołuje skrypt restartujący proces lub demon. Do podstawowych cech demona keepalive można zaliczyć:

- Wprowadzenie interfejsu wiersza poleceń (spawndaemon) z wieloma opcjami dla restartowania procesów/demonów.
- Monitorowanie procesów czasu rzeczywistego.
- Monitorowanie demonów.
- Restartowanie procesów/demonów na dowolnym węźle w klastrze, w sposób określony przez użytkownika.

Demon keepalive używa standardowych skryptów powłoki do restartowania procesów/demonów, które zostały przerwane.

Jak widać zapewnienie dostępności demona zarządzającego można zrealizować konfigurując odpowiednio demony keepalive i spawndeamon. W przypadku braku możliwości skorzystania z tych demonów może jednak być zastosowana jedna z metod opisanych we wcześniejszych opracowaniach projektowych [Naj], [Ost], [Maz] z ewentualnymi autorskimi modyfikacjami. Innym rozwiązaniem, odpowiednim dla aktualnej konfiguracji środowiska laboratoryjnego jest uruchamianie procesu demona na *węźle inicjującym*. Przy obecnej konfiguracji awaria tego węzła i tak powoduje niedostępność całego klastra.

Zakończenie pracy demona Podczas zamykania demona zarządzającego zamykane są wszystkie usługi uruchomione w wyniku jego działania. Jest to istotne ze względów zachowania porządku w środowisku OpenSSI.

Uwagi końcowe Powyższa koncepcja zakłada uruchomienie procesów węzłów MySQL Cluster na konkretnej maszynie, identyfikowanej adresem IP zawartym w plikach konfiguracyjnych. Należy więc zapewnić, że procesy te nie będą migrować wewnątrz klastra. Jedynym procesem, który nie tylko może, ale wręcz powinien migrować w razie awarii komputera na którym działa jest opisany demon zarządzający. Alternatywą jednak jest uruchomienie demona na węźle inicjującym, który w klastrze i tak jest pojedynczym punktem awarii.

7.4.3 Składowanie danych na dysku twardym

MySQL Cluster, począwszy od wersji 5.1.6 umożliwia przechowywanie nieindeksowanych kolumn tabel NDB na dysku twardym, zamiast w pamięci RAM, jak to miało miejsce w wypadku wcześniejszych wersji klastra. Konfiguracja składowania danych na dysku odbywa się już na poziomie samej bazy danych. Może więc zostać zrealizowana przy użyciu skryptów SQL. Skrypty takie zostaną opracowane w ramach implementacji koncepcji autorskiego rozwiązania. Ich szczegóły znajdują się w dokumentacji technicznej, zaś ogólny zarys składa się z trzech kroków:

1. Należy stworzyć grupę plików logów i przypisać do niej jeden lub więcej plik logów typu *UNDO*.

2. Należy stworzyć przesteń tabel (ang. *tablespace*), przypisać do niej grupę plików logów UNDO oraz jeden lub więcej plików danych, które będą wykorzystane do przechowywania danych na dysku.

3. Tabele, które mają być przechowywane w przestrzeni tabel dyskowych należy tworzyć dodając do zapytania CREATE TABLE opcje:

   ```
   TABLESPACE ts_1 STORAGE DISK
   ENGINE NDB;
   ```

7.4.4 Przezroczystość rozwiązania i równoważenie obciążenia

Ważną cechą systemu rozproszonego jest to, by dla zwykłego użytkownika był on widoczny jako pojedynczy system świadczący określone usługi. W odniesieniu do projektowanego rozwiązania oznacza to, że rozwiązanie bazodanowe pracujące w środowisku OpenSSI powinno być widoczne dla klientów MySQL jak pojedynczy serwer MySQL. OpenSSI oferuje interesującą z tego punktu widzenia funkcjonalność jaką jest *Cluster Virtual IP Address*. Jest to mechanizm zapewniający widoczność klastra z zewnątrz pod pojedynczym adresem IP. Adres IP przypisany klastrowi musi być inny niż adresy IP komputerów w klastrze. CVIP konfigurowane jest poprzez plik konfiguracyjny `/etc/cvip.conf` . Jest to plik XML zawierający informacje o adresie CVIP, adresach rzeczywistych serwerów, itd. Usługa HA-LVS realizuje przekierowania, co więcej oferując możliwość równoważenia obciążenia poszczególnych serwerów rzeczywistych.

Rozwiązanie to byłoby idealne, jednak z kilku przyczyn nie może być zastosowane. Po pierwsze założeniem projektu jest podłączanie się do serwera MySQL z wewnątrz klastra. Na tym poziomie CVIP nie działa. Po drugie, pliki konfiguracyjne mogą być edytowane wyłącznie przez użytkownika z uprawnieniami root. Wreszcie koncepcja rozwiązania docelowego zakłada uruchamianie węzłów SQL na różnych komputerach. Nie jest więc możliwe podanie z góry założonych

adresów IP do których przekierowywane by były zapytania SQL. Rozwiązanie takie działałoby co najwyżej do pierwszej awarii komputera z działającym węzłem SQL. Oczywiście demon zarządzający opisany wcześniej mógłby edytować również plik konfiguracyjny CVIP. Byłoby to rozwiązanie optymalne, jednak znów niemożliwe do zrealizowania z powodu braku odpowiednich uprawnień w środowisku laboratoryjnym.

Konieczne jest więc opracowanie alternatywnej metody, działającej w dynamicznym środowisku. Autorzy koncepcji opracowali kilka wariantów rozwiązania.

Wariant 1 Pomysł ten zakłada otoczenie standardowego klienta MySQL aplikacją wyszukującą dostępne węzły SQL, najłatwiej poprzez czytanie plików konfiguracyjnych ze znanej lokalizacji. Następnie przy pomocy funkcji OpenSSI lub na podstawie monitorowania liczby procesów uruchomionych na każdym z węzłów-kandydatów wybierany jest najmniej obciążony węzeł. Jego adres podstawiany jest do pliku konfiguracyjnego standardowego klienta MySQL, a sam klient wywołany standardowym poleceniem.

Wariant 2 Inną możliwością rozwiązania tego samego problemu jest wykorzystanie komunikacji sieciowej. Aplikacja może rozgłosić komunikat, który nasłuchiwany będzie przez demon zarządzający. Demon odpowie, zwracając listę aktualnie dostępnych węzłów SQL wraz z poziomem ich obciążenia.

Wariant 3 Trzeci wariant zakłada modyfikację kodu klienta MySQL i wykorzystanie komunikacji międzyprocesowej w przestrzeni procesów środowiska OpenSSI. Odpytywanie opisane wcześniej polegałoby na komunikacji bezpośrednio z procesem demona zarządzającego, niezależnie od tego na którym komputerze pracuje.

Opisane pomysły nie są idealne z punktu widzenia przezroczystości. Konieczna jest ingerencja w kod klienta MySQL. Uniemożliwia to więc wykorzystanie dowolnej standardowej aplikacji klienckiej. Jednak biorąc pod uwagę dynamikę rozwiązania wymaganą jest to pewien kompromis między przezroczystością a wysoką niezawodnością rozwiązania. Co więcej, dołączając dodatkową aplikację zastępującą standardowego klienta MySQL oferujemy pełną funkcjonalność nie wymagając dodatkowych czynności ze strony użytkownika. Podobnie można dostarczyć proste API do wykorzystania w aplikacjach.

UWAGA! Do implementacji wybrany został wariant 1. Sprawdzenie obciążenia węzłów odbywać się będzie przy pomocy polecenia `LOADS` zwracającego obciążenie działających węzłów klastra.

7.5 Przewidywane korzyści z implementacji rozwiązania

Przewiduje się, że implementacja koncepcji autorskiego rozwiązania opartego o MySQL Cluster przyczyni się do wzrostu niezwodności tego rozwiązania, poprzez dynamiczną zmianę konfiguracji i reakcję na występujące awarie. Dzięki implementacji aplikacji otaczającej klienta MySQL

uzyskamy przezroczystość w sposób optymalny uwzględniając dynamiczne środowisko, w którym nie możemy założyć znajomości adresów IP konkretnych węzłów, które w trakcie działania mogą ulegać zmianie. Wreszcie konfiguracja przestrzeni tabel dyskowych rozwiązania MySQL Cluster doprowadzi do zmniejszenia wymogów dotyczących rozmiarów dostępnej pamięci RAM całego rozwiązania. Przyniesie to też dodatkowe korzyści w postaci krótszego czasu przywracania danych.

8 Zestaw testów wydajnościowych i naciskowych

8.1 Cel testów

Cel testowania bazy danych jest zależny od przedmiotu testowania lub, inaczej, od konkretnego aspektu funkcjonowania serwisu oferującego usługi bazodanowe. Dlatego też, testy klasyfikowane są ze względu na zastosowanie. Należy tu zaznaczyć, że poniżej przedstawiona klasyfikacja nie posiada ściśle zdefiniowanych granic. Gruntowne przetestowanie usług bazodanowych pod kątem wydajności dostarczy testującemu wiele cennych informacji dotyczących reakcji serwera na obciążenia. Podobnie, weryfikacja zachowania bazy danych przy dużym wolumenie transakcji, czy zapytań może być traktowana jak test odporności systemu w sytuacjach ekstremalnych. Właściwe, gruntowne przeprowadzenie testów umożliwi wprowadzenie odpowiednich zmian konfiguracyjnych, celem optymalizacji wydajności bazy i zwiększenia jej niezawodności oraz odporności na awarie. Wyróżniamy kilka podstawowych rodzajów testów baz danych.

- Testy wydajnościowe - Zwane są również profilowaniem wydajnościowym. Ich celem jest weryfikacja czasów odpowiedzi dla wyznaczonych transakcji, zbiorów transakcji, czy funkcji biznesowych przy zachowaniu obciążenia serwerów na względnie stałym, przewidywalnym i określonym poziomie tak, aby nie przekroczyć z góry znanego najwyższego obciążenia.

- Testy obciążeniowe - Ich celem jest zbadanie czasów reakcji i odpowiedzi serwera na zapytania klientów dla wyznaczonych transakcji, zbiorów transakcji, czy funkcji biznesowych przy zróżnicowanym obciążeniu. Przeprowadzenie testów obciążeniowych pozwala określić zależność czasu reakcji od obciążenia serwera lub serwerów w przypadku systemu klastrowego.

- Testy naciskowe - Celem testów naciskowych jest zbadanie odporności systemu bazodanowego na możliwe sytuacje wyjątkowe. Innymi słowy, jest to weryfikacja zachowania funkcjonalności i prawidłowości działania serwera bazy danych w nietypowych sytuacjach, takich jak niedobór pamięci operacyjnej, maksymalna ilość klientów, problemy z połączeniem sieciowym, restart procesu serwera, itd.

- Testy objętościowe - W pewnym sensie są podzbiorem testów naciskowych. Ich rola sprowadza się do weryfikacji systemu bazodanowego pod kątem prawidłowego funkcjonowania przy możliwych do zaistnienia ekstremalnych ilościach zapytań, na jakie serwer jest w stanie odpowiedzieć. Scenariusze takie realizuje się dla maksymalnej, możliwej ilości klientów prawdziwych, bądź symulowanych programowo, przeprowadzających najbardziej niekorzystne dla systemu funkcje biznesowe, zbiory zapytań lub transakcji w nietypowo długim czasie. Dodatkowo, testy realizuje się dla maksymalnie dużej wielkości bazy danych.

Przy realizacji projektu klastrowej bazy danych zdecydowaliśmy się skupić na aspekcie odpornościowym systemu. Jednakże, istotna jest również weryfikacja wydajności zastosowanych rozwiązań. Dlatego też, głównym celem testów będzie weryfikacja odporności systemu na sytuacje wyjątkowe oraz porównanie wydajności i czasów reakcji zwykłego, pojedynczego serwera

z czasami reakcji rozwiązania klastrowego - zarówno gotowego, proponowanego przez programistów MySQL, jak i naszego, opisanego w niniejszej dokumentacji. W związku z powyższym, kolejne, dwie sekcje dokumentacji skupiają się na rozwinięciu idei testów wydajnościowych oraz naciskowych, weryfikujących odporność systemu.

8.2 Testy wydajnościowe

Celem profilowania wydajnościowego jest weryfikacja czasów odpowiedzi serwera bazodanowego dla wyznaczonych transakcji przy względnie stałym obciążeniu serwera. Testowane bazy danych powinny cechować się odpowiednią wielkością. Mała baza, posiadająca niewiele rekordów, w normalnych warunkach, zazwyczaj cechuje się niezwykle wysoką wydajnością. Jednak, jeśli chcemy, aby nasza usługa bazodanowa była skalowalna, powinniśmy przeprowadzić testy dla odpowiednio dużej ilości danych.

Często stosowaną techniką przeprowadzania tego typu testów jest modyfikacja lub generacja zbiorów danych pozwalająca zwiększyć ilość transakcji, mogących mieć miejsce, oraz zastosowanie odpowiednich skryptów testujących, które umożliwiają wykonanie wielu zapytań i symulację określonej ilości klientów. Testy wydajnościowe przeprowadza się iteracyjnie, co pozwala oszacować wydajność systemu względnie precyzyjnie i zminimalizować błędy. Zazwyczaj wyniki testów wydajnościowych są uśrednione, dlatego też duża ilość iteracji jest pożądana. Testy wydajnościowe przeprowadza sie dla jednego bądź wielu wirtualnych lub rzeczywistych klientów.

Niezwykle istotny jest fakt, iż podczas trwania testów wydajnościowych obciążenie systemu powinno być utrzymywane na stałym poziomie. Istnieje kilka metod służących spełnieniu tego wymagania. Jedną z nich jest zastosowanie idei wirtualnych użytkowników do symulacji dużej liczby klientów. Często stosuje się nawet do kilkuset wirtualnych klientów. Dostępne są specjalne narzędzia służące symulowaniu pożądanej liczby użytkowników bazy danych. Godną uwagi jest również idea testowania wydajnościowego przeprowadzanego w warunkach produkcyjnych. Obciążenie systemu jest wówczas naturalne - niestety nie zawsze jest stałe, co może mieć znaczący wpływ na dokładność pomiarów. Z tego względu emulatory obciążenia mają przewagę nad testowaniem w warunkach naturalnych.

Kryteria kompletności testów wydajnościowych nie są skomplikowane. Wykonanie skryptów testowych powinno zakończyć się w zdefiniowanym czasie, przeznaczonym na wykonanie określonych zbiorów transakcji. Oczywiście, realizacja transakcji powinna mieć bezbłędny przebieg.

W Internecie dostępnych jest wiele interesujących programów i środowisk o otwartych źródłach, służących to przeprowadzania testów wydajnościowych. Przy realizacji projektu zdecydowaliśmy się zastosować dwa z takich narzędzi. Koncepcja ich zastosowania wraz z wymaganymi modyfikacjami jest opisana w dalszej części dokumentacji.

8.3 Testy naciskowe

Celem testów naciskowych jest zbadanie odporności systemu bazodanowego na przewidywane sytuacje wyjątkowe. Funkcjonalność testowanego systemu powinna działać prawidłowo dla niety-

powych sytuacji, zarówno skrajnych pod względem obciążenia, jak i awaryjnych. Przy realizacji testów naciskowych z powodzeniem można stosować techniki testów obciążeniowych. Podczas przeprowadzania testów naciskowych warto rozważyć zweryfikowanie odporności systemu bazodanowego dla niżej przedstawionych sytuacji.

- Całkowity brak lub niedobór pamięci operacyjnej na maszynie, na której działa proces serwera - Należy pamiętać, iż w przypadku rozwiązania MySQL Cluster mamy do czynienia z wieloma rodzajami współpracujących serwerów - serwery zapytań, serwery udostępniające dane oraz proces odpowiedzialny za zarządzanie klastrem. Toteż, istotny jest niedobór pamięci operacyjnej maszyn, na których działają wszystkie rodzaje serwerów.

- Maksymalna, możliwa liczba podłączonych do bazy danych klientów - Należy zauważyć, iż w przypadku rozwiązania klastrowego, system powinien tolerować dużo większą ilość klientów, niż standardowa baza danych, działająca w oparciu o jeden serwer. Warto rozważyć zastosowanie mechanizmów symulowania wirtualnych klientów.

- Wielu użytkowników przeprowadzających transakcje dla tych samych danych - Rozwiązanie klastrowe, dysponujące mechanizmami równoważenia obciążeń i co najmniej kilkoma replikami danych, powinno być testowane zarówno dla wielu, jak i dla jednej działającej repliki.

- Najbardziej niekorzystny, dopuszczalny wolumen zapytań i transakcji - Podobnie jak w powyższym przypadku, rozwiązanie klastrowe, z założenia, powinno tolerować znacznie większe ilości symultanicznych transakcji. Należy to uwzględnić w trakcie realizacji testów. Dla dobrze skonfigurowanego rozwiązania klastrowego testy powinny wykazać dużo większą wydajność systemu w porównaniu do tradycyjnego rozwiązania.

- Odcięcie dostępu do skrajnej, dopuszczalnej ilości procesów serwerowych - Oczywiście, tego rodzaju test ma sens tylko dla rozwiązania klastrowego. Należy uwzględnić każdy rodzaj serwerów. Odcięcie dostępu można symulować poprzez zabicie procesu lub odcięcie serwera od klastra.

- Podniesienie wcześniej zabitego procesu serwerowego lub naprawienie połączenia sieciowego - Niezwykle istotna jest obsługa sytuacji, w której, wcześniej odcięty, bądź zabity proces, pojawia się i próbuje kontynuować współpracę z systemem klastrowym. Testy naciskowe powinny zweryfikować zachowanie systemu w takiej sytuacji.

Odporność na na nietypowe sytuacje jest szczególnie istotna w systemach rozproszonych. Fakt rozproszenia zasobów i procesów między wiele maszyn powoduje powstanie dodatkowych zagrożeń dla skuteczności i niezawodności systemu. Uruchomienie rozwiązania MySQL Cluster w systemie operacyjnym OpenSSI wymaga potraktowania aspektów odpornościowych systemu w sposób wyjątkowy. Wydajność osiągnięta przez programistów MySQL jest wysoka i poparta latami pracy i doświadczenia, jak również weryfikacją działania bazy danych w środowiskach produkcyjnych. Podejście klastrowe w rozwoju baz danych jest wciąż rozwijane i wiele aspektów odpornościowych wymaga testów, szczególnie po zainstalowaniu w nietypowym dla rozwiązania środowisku, jakim jest OpenSSI. Bardzo istotna jest weryfikacja odporności klastrowej bazy danych na awarie wynikające z rozproszenia zasobów. Większość testów naciskowych, w

szczególności obciążeniowe, można wykonać za pomocą tych samych narzędzi, które służą do profilowania wydajnościowego. Idea przeprowadzenia testów naciskowych na zainstalowanej rozproszonej bazie danych, jak również na jej zmodyfikowanej wersji, przedstawiona jest w dalszej części dokumentacji.

8.4 Narzędzia

Niniejsza sekcja dokumentacji koncentruje się na przedstawieniu wybranych narzędzi do testowania baz danych. Podane są zalety i wady każdego z ich w kontekście przeprowadzenia wymaganych testów, jak rówiez motywacja przy wyborze konkretnych narzędzi i odrzuceniu pozostałych. Na końcu znajduje się propozycja autorskiej modyfikacji wybranych narzędzi tak, aby dostosować je do naszych potrzeb.

8.4.1 Narzędzia testujące MySQL

Dostępnych jest wiele narzędzi umożliwiających przeprowadzenie gruntownych testów wydajnościowych systemu bazodanowego działającego w oparciu o MySQL. Narzędzia te z powodzeniem mogą również posłużyć do wykonania większości testów naciskowych. Bez potrzeby pisania własnej aplikacji od zera, użytkownik może za pomocą tych narzędzi wygenerować odpowiednie dane, na których następnie zostaną wykonane zautomatyzowane testy, przeprowadzane przez gotowe skrypty. Środowiska służące do testowania baz danych są zazwyczaj wyposażone w dużą ilość konfigurowalnych parametrów, co pozwala dostosować testy do potrzeb użytkownika. Poniżej znajdują się krótkie opisy niektórych, dostępnych narzędzi. Dwa ostatnie narzędzia opisane są nieco szerzej, ponieważ to właśnie one zostały wybrane do implementacji środowiska testowego w ramach projektu.

- Benchmark Suite - Narzędzie udostępniane w źródłowych dystrybucjach MySQL'a, które można pobrać ze strony `http://dev.mysql.com/downloads/`. Narzędzie ma na celu sprawdzenie i powiadomienie użytkownika o wydajności operacji wykonywanych na konkretnej implementacji SQL. Narzędzie działa w trybie jednowątkowym, więc mierzy minimalny czas wykonywania operacji. W przyszłości planowane jest dodanie wsparcia dla wielowątkowości.

- OSDB - Narzędzie o otwartych źródłach, zbudowane na podstawie AS3AP (ANSI SQL Standard and Portable Benchmark) z myślą o projektantach baz danych i systemów bazodanowych. OSDB dostarcza gotowego kodu, który może być dowolnie modyfikowany i dostosowywany do potrzeb testowania wydajności baz danych. Szczegółowy opis działania, instalacji oraz użycia pakietu znajduje się w katalogu /docs w pakiecie instalacyjnym.

- BenchW - Narzędzie stworzone do porównywania różnych managerów baz danych na potrzeby magazynów danych. BenchW został zaprojektowany do testowania ładowania danych, tworzenia indeksów i badania wydajności zapytań w bazach danych. Głównym założeniem tego narzędzia jest wypełnienie luki na rynku pomiędzy złożonymi i skomplikowanymi narzędziami komercyjnymi a domorosłymi projektami. BenchW ma być prosty, ale zapewniać na tyle rzeczywistych cech, aby być narzędziem przydatnym w testowaniu wydajności baz danych.

- SysBench - Modularne niezależne od platformy oraz wielowątkowe narzędzie testujące wydajność parametrów systemu operacyjnego, mających wpływ na działanie bazy danych przy dużych obciążeniach. Narzędzie zostało napisane z myślą o przeprowadzaniu testów na bazie MySQL, ale w przyszłości ma obsługiwać także inne implementacje baz danych. Głównym założeniem tego projektu jest szybka analiza wydajności systemu bez wcześniejszej skomplikowanej konfiguracji testów a nawet instalacji samej bazy.

- DBT - Zestaw profesjonalnych testów obciążeniowych oraz testowy 'framework' do przeprowadzania tych testów, opracowany przez OSDL (Open Source Development Labs). Testy zostały zaprojektowane z myślą o profesjonalnych zastosowaniach Linuxa w środowiskach produkcyjnych. Dostępne są specjalnie przygotowane testy dla 4 różnych modeli obciążenia systemów bazodanowych. Zaliczają się do nich następujące modele.

  - Database Test 1 - Testuje wydajność transakcyjnych systemów sieciowych. Symulowana jest aktywność użytkowników przeglądających strony www i kupujących produkty w sklepach 'on-line'. Wyniki DBT-1 zawierają między innymi ilość transakcji obsługiwanych na sekundę (TPS), obciążenie procesora, aktywność operacji I/O oraz zużycie pamięci.
  - Database Test 2 - Test dla systemów OLTP. Symuluje dostęp kilku pracowników do bazy hurtowni produktów zmieniających dane klientów oraz sprawdzających informacje o produktach przechowywanych w bazie. Wyniki DBT-2 zawierają między innymi ilość transakcji obsługiwanych na sekundę (TPS), obciążenie procesora, aktywność operacji I/O oraz zużycie pamięci.
  - Database Test 3 - Symuluje obciążenie systemu wspierającego decyzje. Zawiera zestaw zapytań zorientowanych na podejmowanie decyzji biznesowych na podstawie danych w bazie oraz współbieżnie modyfikuje dane w bazie.
  - Database Test 4 - Symuluje obciążenie serwera aplikacyjnego udostępniającego usługi sieciowe (ang. Web services). Test składa się z zadań podobnych do występujących w transakcyjnym środowisku 'business-to-business'.

- mysqlslap - Program diagnostyczny opracowany przez programistów MySQL - jest więc w pełni kompatybilny z bazami danych MySQL. Celem jego działania jest emulacja wielu klientów korzystających z usług bazodanowych tak, aby symulować naturalne obciążenie serwera, celem przetestowania czasów reakcji bazy danych na wszystkich etapach realizacji transakcji - od wygenerowania przez klienta zapytania do otrzymania pełnej, bezbłędnej odpowiedzi. Program posiada bogatą funkcjonalność. Do najistotniejszych cech zaliczają się niżej wymienione elementy.

  - Różnorodne źródła zapytań - Zapytania mogą być podawane w linii poleceń, generowane automatycznie, bądź sczytywane ze specjalnie przygotowanych plików.
  - Konfigurowalna liczba klientów oraz iteracji - Program udostępnia parametry pozwalające określić liczbę wirtualnych klientów, generujących zapytania oraz liczbę iteracji wykonywanych testów, a nawet ilość zapytań dla każdego klienta.

- Wyjście do pliku CSV - Istnieje możliwość zapisania wyników testów w pliku o formacie obsługiwanym przez arkusze kalkulacyjne. Jest to niezwykle istotne ze względu na możliwość przeprowadzenia analizy otrzymanych rezultatów.
- Definicja serwera - Przy realizacji testów w środowiskach istnieje możliwość zdefiniowania hosta, na którym zlokalizowany jest serwer bazy danych, co ma szczególne znaczenie w przypadku rozproszenia procesów.
- Konfiguracja - Program wyposażony jest w bardzo wiele opcji konfiguracyjnych, pozwalających dostosować go do konkretnej bazy danych.

Każde z wyżej opisanych narzędzi posiada zalety i wady. Kierując się własnymi wymaganiami oraz możliwościami programów dokonaliśmy selekcji i wybraliśmy dwa środowiska testujące. Kolejny paragraf skupia się na wyjaśnieniu motywacji naszych decyzji dotyczących wyboru odpowiedniego narzędzia. Wartym podkreślenia jest fakt, iż dobrze zaprojektowane, 'parametryzowalne' narzędzie do przeprowadzania profilowania wydajnościowego może z powodzeniem posłużyć do realizacji znaczącej części testów naciskowych. Badanie maksymalnej, możliwej liczby podłączonych klientów, czy najbardziej niekorzystny wolumen zapytań i transakcji mogą być zrealizowane np. za pomocą narzędzia 'mysqlslap'. Pozostałe testy naciskowe, jak np. badanie pracy systemu bazodanowego przy ograniczonym dostępie do medium sieciowego, czy badanie reakcji bazy na odcięcie dostępu do skrajnej ilości procesów serwerowych powinno zostać zweryfikowane innymi metodami. Proponowane przez nas rozwiązania opisane są w dalszej części niniejszego opracowania, natomiast szczegółowe opisy wyżej wymienionych narzędzi, poza 'mysqlslap', znajdują się w załączniku A. Bogatym źródłem informacji odnośnie 'mysqlslap' jest strona internetowa `http://dev.mysql.com/doc/refman/5.1/en/mysqlslap.html`.

8.4.2 Wybrane narzędzie

Po przeanalizowaniu zalet i wad każdego z opisanych narzędzi w kontekście realizacji wiarygodnych testów wydajnościowych i naciskowych zdecydowaliśmy się wybrać narzędzia 'mysqlslap' oraz Database Test 2. Połączenie zalet obu rozwiązań pozwoli nam stworzyć odpowiedni do naszych celów zestaw testów. 'Mysqlslap' jest niezwykle elastycznym programem, wyposażonym w dużą liczbę parametrów pozwalających dostosować jego działanie do potrzeb użytkownika. Ponadto, jest dedykowanym narzędziem do testowania baz danych MySQL, które wspiera wielowątkowość. DBT-2, z kolei, jest w pełni zautomatyzowanym silnikiem do przeprowadzania testów. Skrypty, w oparciu o które działa, pozwalają na względnie łatwą modyfikację. DBT-2 nie wspiera jednak wielowątkowości, toteż wprowadzenie odpowiednich zmian w kodzie skryptów, pozwalających połączyć dwa narzędzia, umożliwi przeprowadzenie odpowiednich testów.

Pozostałe narzędzia zostały wykluczone, ponieważ nie spełniają odpowiednich wymagań. Poniżej przedstawiono argumenty, które przemówiły za odrzuceniem poszczególnych narzędzi testujących.

- Benchmark Suite - Mimo, iż narzędzie udostępniane jest w źródłowych dystrybucjach MySQL'a, posiada jedną dużą wadę - na dzień dzisiejszy nie wspiera wielowątkowości. Nie można więc za jego pomocą symulować dużej liczby klientów symultanicznie pracujących z daną bazą danych, a przeprowadzenie tego rodzaju testu jest szczególnie istotne podczas projektowania rozwiązania klastrowego.

- OSDB - Narzędzie o otwartych źródłach, dostarczające gotowego kodu, który może być dowolnie modyfikowany i dostosowywany do potrzeb testowania wydajności baz danych. Niewątpliwie jest to narzędzie godne uwagi. Niestety, wspiera maksymalnie MySQL 3.23 i nie może zostać zastosowane no naszych celów, gdyż projekt realizowany jest w oparciu najnowszą, na dzień dzisiejszy, dostępną wersję, jaką jest 5.1.6.
- BenchW - Jest narzędziem stworzonym do porównywania managerów baz danych na potrzeby magazynów i hurtowni danych. Nam jednak zależy na przetestowania bazy danych o standardowych rozmiarach. Tym bardziej, że rozwiązanie MySQL Cluster, póki co, nie wspiera przechowywania danych na dysku - przechowuje je w pamięci operacyjnej, co mocno ogranicza jego zastosowanie przy dużych ilościach danych.
- SysBench - Jest narzędziem testującym wydajność parametrów systemu operacyjnego, mających wpływ na działanie bazy danych przy dużych obciążeniach. Zespoły projektujące rozwiązanie bazodanowe w ramach projektu RSO nie mają wpływu na działanie systemu operacyjnego. Ponadto, konfiguracja systemu nie jest celem projektu.

8.4.3 Autorskie modyfikacje narzędzia

W celu sprawdzenia wydajności serwera MySQL posłużyliśmy się narzędziem mysqlslap. Jest to standardowe narzędzie diagnostyczne symulujące obciążenie klienckie serwera MySQL. Jego działanie polega na symulacji połączenia się wielu klientów do serwera i zebraniu czasów poszczególnych obciążeń serwera.

Jako bazę testującą wybraliśmy bazę tworzoną przy okazji instalacji innego narzędzia diagnostycznego OSDL DBT-2. W obydwu przypadkach będziemy wykorzystywać ten sam zbiór zapytań. Po przeprowadzeniu wszystkich testów wyniki zostaną porównane.

Skrypt wykorzystujący narzędzie *'mysqlslap'*, znajduje się w pliku **mysqlslap_test_script.sh**.

```
#!/bin/sh
usage() {

  if [ "$1" != "" ]; then
    echo ''
    echo "error: $1"
  fi

  echo ''
  echo 'usage: mysqlslap_test_script.sh'
  echo 'options:'
  echo '       -d <database name>'
  echo '       -o <output file name>'
  echo '       -c <path to mysqlslap client binary.
```

```
(default: $HOME/mysql/bin/mysqlslap)'
  echo '       -s <database socket>'
  echo '       -h <database host (default: localhost)'
  echo '       -i <input file for mysqlslap, that
include queries>'
  echo '       -n <numer of clients>'
  echo 'Example: ./mysqlslap_test_script.sh -d
dbt2 -n 10 -i input.sql -o explain.out'
  echo ''
}

NUM_CLIENT=1

while getopts "o:d:c:s:h:i:n:" opt; do
        case $opt in
        o)
                OUTPUT_FILE=$OPTARG
                ;;
    i)
        INPUT_FILE=$OPTARG
        ;;
        d)
                DB_NAME=$OPTARG
                ;;
    c)
        MYSQL=$OPTARG
        ;;
    s)
        DB_SOCKET=$OPTARG
        ;;
    h)
        DB_HOST=$OPTARG
        ;;
    n)
        NUM_CLIENT=$OPTARG
        ;;
    ?)
        usage
        exit 1
        ;;
        esac
done

if [ "$OUTPUT_FILE" == "" ]; then
   usage "specify ouput filename -o <outfile>"
```

```
   exit 1
fi

if [ "$INPUT_FILE" == "" ]; then
   usage "specify input filename -i <infile>"
   exit 1
fi

if [ "$MYSQL" == "" ]; then
  MYSQL="$HOME/mysql/bin/mysqlslap"
fi

if [ "$DB_SOCKET" == "" ]; then
  DB_SOCKET="$HOME/mysql/thesocket"
fi

if [ "$DB_HOST" == "" ]; then
  DB_HOST="localhost"
fi

if [ "$DB_NAME" == "" ]; then
  usage "specify database name using -d #"
  exit 1
fi


MYSQL="$MYSQL -v"

echo "---------NUMBER OF CLIENTS: $NUM_CLIENT;
 INPUT FILE: $INPUT_FILE --------">> $OUTPUT_FILE

$MYSQL $DB_NAME -S $DB_SOCKET -u root -c
$NUM_CLIENT -q $INPUT_FILE >> $OUTPUT_FILE

echo "---- FINISH ----" >> $OUTPUT_FILE
```

Opis sposobu użycia skryptu wygląda następująco:

```
usage: mysqlslap_test_script.sh'
options:
        -d <database name>
        -o <output file name>
        -c <path to mysqlslap client binary.
(default: HOME/mysql/bin/mysqlslap)
        -s <database socket>
        -h <database host (default: localhost)
```

```
        -i <input file for mysqlslap,
that include queries>
        -n <numer of concurrency clients>
Example: ./mysqlslap_test -d dbt2 -n 10 -i input.sql
        -o outtemp.out
```

Wywołanie skryptu wymaga podania nazwy bazy danej, ilości równoległych połączeń klienckich, nazwę wejściowego pliku zawierającego zapytania typu SELECT oraz nazwę pliku wyjściowego. Po pomyślnym wykonaniu testu, wyniki znajdują się w pliku wyjściowym. Wynik pojedynczego testu wygląda następująco:

```
------NUMBER OF CLIENTS: 200; INPUT FILE: mysqlslap_input -------
Benchmark
      Average number of seconds to run all queries: 0.048 seconds
      Minimum number of seconds to run all queries: 0.048 seconds
      Maximum number of seconds to run all queries: 0.048 seconds
      Number of clients running queries: 200
      Average number of queries per client: 35

-------------------------FINISH---------------------------------
```

8.5 Procedury testowania

Ponizsza sekcja dokumentacji skupia się na przedstawieniu proponowanych procedur testowania. Zdefiniowane są przedmioty testów oraz kryteria porównywania wyników. Następnie opisane są propozycje testów wydajnościowych oraz naciskowych. Na końcu sekcji znajduje się opis sposobu prezentacji i porównywania wyników.

8.5.1 Przedmioty testowania

Każdy 'kamień milowy', zdefiniowany w temacie projektu RSO, kończy się uruchomieniem odpowiedniej usługi bazodanowej. Pod koniec pierwszego etapu uruchomiono tradycyjną usługę MySQL działającą w oparciu o jeden serwer. Celem drugiego etapu jest uruchomienie klastrowego rozwiązania MySQL Cluster. Ostatni etap zakończy się zaimplementowaniem pewnych zmian klastrowej bazy danych, opisanych w niniejszej dokumentacji.

Przejście od rozwiązania prezentowanego w pierwszym etapie do rozwiązania klastrowego, ma na celu zwiększenie wydajności i możliwości systemu, jak również zapewnienie większej niezawodności poprzez wprowadzenie redundancji zasobów i procesów. Dlatego też, warto przeprowadzić wydajnościowe testy porównawcze pojedynczego serwera i rozwiązania klastrowego. Niewątpliwie, rozproszona baza danych powinna być szybsza i bardziej wydajna. Powinna również tolerować większe obciążenia, wynikające zarówno ze zwiększonej ilości symultanicznie pracujących klientów, jak i niekorzystnego wolumenu transakcji.

Rozwiązanie klastrowe stworzone przez przez programistów MySQL jest wciąż rozwijane. Projekt jest zaawansowany, jednak wciąż wiele jest do zrobienia. W ramach realizacji projektu zdecydowaliśmy się zwiększyć odporność systemu na awarie oraz wprowadzić przezroczystość pomiędzy klientem bazy a grupą serwerów zapytań. Proponowane przez nas zmiany będą miały wpływ nie tylko na odporność systemu, ale również na jego wydajność. Należy więc dokonać gruntownych testów, zarówno wydajnościowych, jak i naciskowych, weryfikujących poprawność proponowanych przez nas rozwiązań. Zalety naszego autorskiego rozwiązania powinny zostać porównane z zaletami rozwiązania wyjściowego, jakim jest MySQL Cluster.

8.5.2 Testy wydajnościowe - opis procedur

Celem testów wydajnościowych jest weryfikacja czasów odpowiedzi dla wyznaczonych transakcji, zbiorów transakcji, czy funkcji biznesowych przy zachowaniu obciążenia serwerów na względnie stałym, przewidywalnym i określonym poziomie tak, aby nie przekroczyć z góry znanego najwyższego obciążenia.

Testy przeprowadzane mogą być na dwa sposoby. Pierwszym z nich jest pomiar czasu wykonania transakcji. Procedury testów przewidują tu porównanie czasów wykonania testów dla emulacji obciążenia 700, 1000, 1300, 1500, 1800 oraz 2000 klientów. Obciążenie symulowane jest przy wykorzystaniu aplikacji 'mysqlslap'.

Drugi rodzaj testów polega na założeniu maksymalnego czasu wykonania transakcji i zbadanie jak wiele prób zakończy się sukcesem, a więc wykonaniem transakcji w czasie krótszym. Testy przeprowadzone są, podobnie jak w poprzednim wypadku, przy użyciu emulatora 'mysqlslap' 700, 1000, 1300, 1500, 1800 oraz 2000 klientów.

Wszystkie testy przeprowadzane są na testowej bazie danych pochodzącej z aplikacji DBT-2.

8.5.3 Testy naciskowe - opis procedur

Podczas przeprowadzania testów naciskowych należy zweryfikować odporność systemu dla niżej przedstawionych sytuacji. Część testów może być wykonana za pomocą narzędzi do profilowania wydajnościowego. Jeżeli tylko istnieje taka możliwość, jest to wyraźnie podkreślone w poniższych opisach. Dokładne liczby symultanicznie działających klientów, czy generowanych zapytań zostaną określone doświadczalnie podczas przeprowadzania testów. Ważną rolą testów naciskowych jest zbadanie granic wytrzymałości systemu bazodanowego. Wyniki takich testów mogą posłużyć nie tylko do weryfikacji odporności systemu, ale również do korekcji ustawień konfiguracyjnych.

- Całkowity brak lub niedobór pamięci operacyjnej na maszynie, na której działa serwer - W przypadku rozwiązania klastrowego istotny jest niedobór pamięci operacyjnej maszyn, na których działają wszystkie rodzaje serwerów. Należy pamiętać, iż w rozwiązaniu MySQL Cluster, dane przechowywane są w pamięci operacyjnej, toteż w przypadku serwerów replik, dostateczna ilość pamięci RAM maszyn, na których owe serwery działają, jest kluczowa. Niedobór pamięci operacyjnej można symulować np. za pomocą narzędzi emulujących dużą liczbę aplikacji klienckich, jak 'mysqlslap'.

- Maksymalna, możliwa liczba podłączonych do bazy klientów - Należy zauważyć, iż w przypadku rozwiązania klastrowego, system powinien tolerować dużo większą ilość klientów, niż standardowa baza danych. Warto rozważyć zastosowanie mechanizmów symulowania wirtualnych klientów przez wyżej wymieniony, wielowątkowy 'mysqlslap'.

- Wielu użytkowników przeprowadzających transakcje dla tych samych danych - Rozwiązanie klastrowe, dysponujące mechanizmami równoważenia obciążeń i co najmniej kilkoma replikami danych, powinno być testowane dla dużej liczby klientów. Możliwość symulacji symultanicznych zapytań, generowanych przez wiele klientów, jest zapewniona przez program 'mysqlslap'.

- Najbardziej niekorzystny, dopuszczalny wolumen zapytań i transakcji - Znowu, rozwiązanie klastrowe, z założenia, powinno tolerować znacznie większe ilości symultanicznych transakcji. Testowanie obsługi krytycznych ilości zapytań z powodzeniem może zostać zrealizowana za pomocą narzędzia 'mysqlslap'.

- Odcięcie dostępu do skrajnej, dopuszczalnej ilości procesów serwerowych - Tego rodzaju test ma sens tylko dla rozwiązania klastrowego. Należy uwzględnić każdy rodzaj serwerów działających w systemie MySQL Cluster. Odcięcie dostępu można symulować poprzez zabicie procesu lub odcięcie serwera od klastra (poprzez zmianę ustawień TCP/IP lub fizyczne odłączenie kabla). Niezwykle istotna jest obsługa sytuacji, w której, wcześniej odcięty, bądź zabity proces, pojawia się i próbuje kontynuować współpracę z systemem klastrowym.

- Niekorzystny wolumen ruchu sieciowego - Aplikacje działające w architekturze klient-serwer bardzo często wymagają sprawnie działającej sieci komputerowej. O ile w przypadku tradycyjnej bazy danych często stosuje się klienty uruchamiane na tych samych maszynach co serwery, o tyle w przypadku klastrowej bazy danych rozproszenie zasobów jest naturalne i, z założenia, oczywiste. Warto więc zweryfikować zachowanie systemu rozproszonego przy niedostępności lub ograniczeniu dostępu do medium sieciowego. Obciążenie sieci lokalnej może być symulowane poprzez generowanie kontrolowanych rozgłoszeń.

9 Wyniki testów wydajnościowych

9.1 Standardowa instalacja MySQL Server

W testach została wykorzystana baza danych, która jest podstawą programu testującego OSDL Database Test 2 (w skrócie DBT-2). Po stworzeniu bazy na serwerze MySQL, skrypt testujący programu DBT-2 został tak zmodyfikowany, aby w testach wykorzystywał program mysqlslap. Zestaw zapytań sql'owych, na postawie których jest wyliczana wydajność serwera MySQL pochodzi również z wspomnianego DBT-2. Poniżej, zostały umieszczone wyniki testów.

NUMBER OF CLIENTS: 700 Benchmark Average number of seconds to run all queries: 0.261 seconds Minimum number of seconds to run all queries: 0.261 seconds Maximum number of seconds to run all queries: 0.261 seconds Number of clients running queries: 700 Average number of queries per client: 36

NUMBER OF CLIENTS: 1000 Benchmark Average number of seconds to run all queries: 0.390 seconds Minimum number of seconds to run all queries: 0.390 seconds Maximum number of seconds to run all queries: 0.390 seconds Number of clients running queries: 1000 Average number of queries per client: 36

NUMBER OF CLIENTS: 1300 Benchmark Average number of seconds to run all queries: 0.609 seconds Minimum number of seconds to run all queries: 0.609 seconds Maximum number of seconds to run all queries: 0.609 seconds Number of clients running queries: 1300 Average number of queries per client: 36

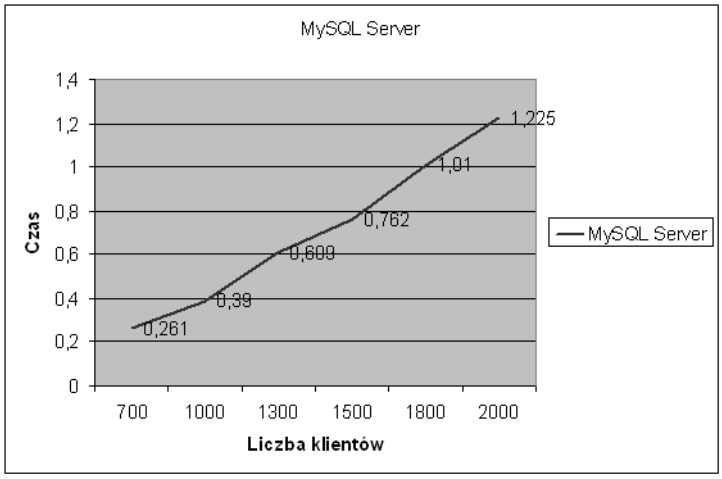
NUMBER OF CLIENTS: 1500 Benchmark Average number of seconds to run all queries: 0.762 seconds Minimum number of seconds to run all queries: 0.762 seconds Maximum number of seconds to run all queries: 0.762 seconds Number of clients running queries: 1500 Average number of queries per client: 36

NUMBER OF CLIENTS: 1800 Benchmark Average number of seconds to run all queries: 1.010 seconds Minimum number of seconds to run all queries: 1.010 seconds Maximum number of seconds to run all queries: 1.010 seconds Number of clients running queries: 1800 Average number of queries per client: 36

NUMBER OF CLIENTS: 2000 Benchmark Average number of seconds to run all queries: 1.225 seconds Minimum number of seconds to run all queries: 1.225 seconds Maximum number of seconds to run all queries: 1.225 seconds Number of clients running queries: 2000 Average number of queries per client: 36

9.2 MySQL z rozszerzeniami klastrowymi

Nie udało się przeprowadzić testów porównawczych dla instalacji MySQL z rozszerzeniami klastrowymi. Powodem był brak możliwości uruchomienia wszystkich usług tej instalacji. W wyniku stworzenia w środowisku bardzo dużej liczby procesów węzłów klastra przez wszystkie



Rysunek 5: Wyniki testów wydajnościowych. Podano czasy wykonania testowych transakcji dla emulowanego obciążenia 700, 1000, 1300, 1500, 1800 oraz 2000 klientami.

grupy projektowe, mimo wielu prób i długiego oczekiwania usługi węzłów danych opracowanej instalacji nie uruchomiły się podczas przeprowadzania testów.

A Algorytmy realizacji wysokiej dostępności

Na samym początku warto się przyjrzeć definicji kluczowych słów, występujących w temacie tego opracowania.

Dostępność *(ang. availability)* definiujemy jako gotowość systemu do natychmiastowego użycia. Ogólnie określamy ją jako prawdopodobieństwo, że system działa poprawnie w dowolnej chwili i jest gotów wykonywać swoje funkcje na zlecenie jego użytkowników. Innymi słowy, system o dużej dostępności to system, który najprawdopodobniej będzie działał w wybranej chwili.

Niezawodność *(ang. reliability)* to zdolność systemu do ciągłego, bezawaryjnego działania. W przeciwieństwie do dostępności niezawodność określamy za pomocą przedziału czasu, a nie chwili. System o dużej niezawodności będzie najprawdopodobniej działał nieprzerwanie w dość długim okresie. Jeśli system wyłącza się na jedną milisekundę co godzinę, to jego dostępność wynosi ponad $99,9999\%$, lecz mimo jest on bardzo zawodny. Podobnie system, który nie załamuje się nigdy, lecz jest wyłączany każdego sierpnia na 2 tygodnie, ma dużą niezawodność, lecz jego dostępność wynosi tylko 96%. Nie są to więc takie same pojęcia.

Wydajność *(ang. efficiency)* - mówimy o systemie, że jest wydajny, w momencie gdy ilość operacji wykonanych w przyjętym przedziale czasie jest zadawalająca, oraz akceptowalna w ramach realizacji konkretnego projektu.

Replika danych stanowi kopie całości lub jakiejś części danych przechowywanych w innym miejscu niż oryginał.

Zwiększona stabilność i dostępność są najczęściej wymienianymi zaletami rozproszonych baz danych. Ulepszenie stabilności i dostępności jest możliwe dzięki wprowadzeniu replikacji danych i oprogramowania. W architekturze zcentralizowanej pojedyncza awaria powoduje niedostępność całego systemu. W architekturze rozproszonej niektóre dane mogą być niedostępne, ale z pozostałych części użytkownicy mogą korzystać bez przeszkód.

Rozproszony system baz danych dzieli bazę danych, przechowując dane bliżej miejsca, gdzie są potrzebne. Lokalizacja danych redukuje zapotrzebowanie na moc obliczeniową i operacje wejścia/wyjścia, zmniejszając jednocześnie opóźnienie w dostępie do danych spowodowane ich przesyłaniem przez sieć rozległą. Kiedy duża baza danych jest rozpraszana na wiele oddziałów, w każdym z nich tworzona jest mała baza danych. W wyniku tej operacji lokalne zapytania i transakcje dotyczą mniejszych baz danych i są przez to obsługiwane szybciej. Ponadto, każdy oddział przetwarza w danym momencie mniej transakcji niż w przypadku wykonywania ich wszystkich w jednej centralnej bazie danych. Co więcej, zrównoleglenie interzapytań i intrazapytań poprzez wykonywanie wielu zapytań w różnych miejscach lub równoczesne wykonanie kilku podzapytań prowadzi do dalszego polepszenia wydajności.

A.1 Replikacja

Replikacja zwiększa dostępność danych.

A.1.1 Podział replikacji ze względu na stopień replikacji

Skrajnym przypadkiem replikacji jest przechowywanie kopii całej bazy danych w każdym węźle systemu rozproszonego, tworząc **w pełni replikowaną rozproszoną bazę danych**. Istotnie zwiększa to dostępność, ponieważ cały system pracuje poprawnie, dopóki choć jeden węzeł jest sprawny. Zwiększa to również wydajność znajdowania wyników globalnych zapytań, ponieważ są one przechowywane w lokalnej replice, a zatem wszystkie zapytania mogą być obsługiwane przez lokalny węzeł (o ile ten węzeł zawiera moduł serwera). Wadą pełnej replikacji jest drastyczne spowolnienie przeprowadzania aktualizacji danych, ponieważ dla zachowania spójności każda dana musi zostać zapisana we wszystkich węzłach. Spowolnienie jest szczególnie uciążliwe jeśli jest duża ilość węzłów. Pełna replikacja zwiększa również koszt technik sterowania współbieżnego i odzyskiwania danych.

Drugą skrajnością, przeciwną do pełnej replikacji, jest **brak replikacji**, który oznacza, że każdy fragment jest zapisywany dokładnie w jednym węźle.

Pomiędzy tymi dwiema skrajnościami istnieje szerokie spektrum **częściowej replikacji danych**, w przypadku której niektóre fragmenty bazy danej są replikowane, a inne nie. Liczba kopii każdego fragmentu może się wahać od jednej do całkowitej liczby wszystkich węzłów w systemie. Szczególny przypadek częściowej replikacji jest stosowany w przypadku pracowników mobilnych, takich jak handlowcy, planiści finansowi czy inspektorzy ubezpieczeniowi, którzy przenoszą replikowaną bazę ze sobą na urządzeniach przenośnych typu PDA (ang. Personal Digital Assistant) lub laptopach i synchronizują ją okresowo z serwerem bazy danych.

Wybór węzłów i stopień replikacji zależy od wymaganej wydajności i dostępności systemu oraz rodzaju i częstotliwości transakcji wykonywanych przez każdy z węzłów. Jeżeli na przykład wymagana jest dyspozycyjność systemu, transakcje mogą być wykonywane przez każdy węzeł i większość transakcji polega na odczytywaniu danych, to dobrym rozwiązaniem będzie pełna replikacja. Z drugiej strony, jeżeli transakcje używające określone części bazy danych są głównie wykonywane przez jeden węzeł, to warto umieścić odpowiednie fragmenty tylko w tym węźle. Dane używane w kilku węzłach dobrze jest umieścić właśnie w nich, pamiętając jednak, że w przypadku dokonywania wielu aktualizacji danych warto ograniczyć replikację. Znalezienie optymalnego, a przynajmniej dobrego rozwiązania problemu alokacji rozproszonych danych jest złożonym problemem optymalizacyjnym.

A.1.2 Synchronizacja replik

Są dwie metody synchronizacji (odświeżenia) replik:

- **Synchroniczna replikacja.** Zanim modyfikująca transakcja zostanie zatwierdzona, należy dokonać aktualizacji wszystkich replik (obejmuje zakładanie blokad, wymianę komunikatów w sieci). Wynik odczytywania danych tabeli jest zawsze taki sam dla każdej jej repliki.

- **Asynchroniczna replikacja.** Kopie zmodyfikowanej tabeli są tylko okresowo aktualizowane - metoda znacznie tańsza, ale chwilowo repliki mogą nie być zsynchronizowane z

zmodyfikowana tabelą.

A.1.3 Prawa własności do danych

Prawa własności określają, które węzły mogą modyfikować dane. Podstawowymi typami praw własności są: prawa typu nadrzędny/podrzędny (ang. master/slave), prawa zgodne z przepływem sterowania (ang. workflow) i prawa modyfikacji dla wszystkich (ang. update-anywhere), nazywane czasem prawami równorzędnymi (ang. peer-to-peer) lub replikacją symetryczną.

Nadrzędny/podrzędny W przypadku tego typu praw własności asynchronicznie replikowane dane są przypisane do jednego węzła nazywanego węzłem nadrzędnym lub głównym i mogą być modyfikowane tylko przez ten węzeł.

Zgodne z przepływem sterowania Model praw własności zgodnych z przepływem sterowania, podobnie jak prawa typu nadrzędny/podrzędny, pozwala uniknąć konfliktów modyfikacji danych. Jednocześnie jednak oferuje on bardziej dynamiczny system przyznawania praw własności. Dokładniej, istnieje tutaj możliwość przekazywania praw do modyfikacji replikowanych danych między węzłami. Jednocześnie zachowany jest warunek, iż w każdym momencie tylko jeden węzeł posiada prawa modyfikacji określonego zbioru danych. Typowym przykładem zastosowania praw własności zgodnych z przepływem sterowania jest przetwarzanie zamówień, na które składa się szereg następujących po sobie kroków.

Prawo modyfikacji dla wszystkich (replikacja symetryczna) Cechą obu metod omawianych powyżej jest zasada, iż w każdym momencie tylko jeden węzeł posiada prawo modyfikacji danych, a pozostałe węzły mają prawo tylko do odczytu replik tych danych. W niektórych przypadkach zasada ta jest zbyt restrykcyjna. Model z prawem modyfikacji dla wszystkich tworzy sieć na zasadzie praw równorzędnych (ang. peer-to-peer), w której wszystkie węzły mają prawo modyfikacji replikowanych danych. Umożliwia to autonomiczną pracę węzłów nawet wtedy gdy inne węzły są niedostępne.

A.2 Algorytmy realizacji wysokiej dostępności

Jak zostało wspomniane na samym początku dostępność systemu rozumiemy jako gotowość systemu do natychmiastowego użycia w każdej chwili. Największą wadą systemów scentralizowanych jest fakt, że awaria węzła centralnego powoduje brak możliwość systemu do kontynuacji działania. W rozproszonych systemach baz danych, projektanci muszą zadbać o wysoką dostępność systemu. W momencie awarii, któregoś z węzłów (najczęściej jest to węzeł, który jest koordynatorem) System Zarządzający Bazami Danymi musi odpowiednio na to zareagować, czego rezultatem powinno być zachowanie ciągłości w działaniu całego systemu.

Przyjmijmy, że mamy do czynienia z rozproszonym systemem bazy danych gdzie mamy węzły nadrzędne oraz węzły podrzędne. Różnica między tymi węzłami jest taka, że wszelkie modyfikacje na bazie danych są robione przez węzeł nadrzędny, zwany koordynatorem. Węzły

nadrzędne i podrzędne posiadają pełną replikację bazy danej oraz odpowiednie oprogramowanie. Te dwa elementy powodują, że każdy z węzłów może w każdym momencie przejąć rolę koordynatora. Każdy węzeł posiada numer będący jego priorytetem. Każdy węzeł wie który węzeł jest aktualnie węzłem nadrzędnym oraz które węzły mają większy priorytet od niego. W momencie gdy następuje awaria węzła nadrzędnego system musi sobie poradzić z tym i wybrać inny węzeł do pełnienia roli koordynatora. Do obsłużenia tego przypadku system może użyć jeden z algorytmów elekcji. Rozróżniamy dwa algorytmy elekcji:

- **Algorytm tyrana** *(ang. bully algorithm)*. Węzeł, który zauważy, że węzeł będący koordynatorem przestał odpowiadać na zamówienie, rozpoczyna elekcję. Węzeł W przeprowadza elekcję następująco:

  1. Węzeł **W** wysyła komunikat ELEKCJA do wszystkich węzłów z wyższym priorytetem.
  2. Jeżeli nikt nie odpowiada to węzeł **W** zwycięża w wyborach i staje się koordynatorem i ogłasza o tym fakcie pozostałe węzły.
  3. Jeśli któryś z procesów o wyższym numerze nadeśle odpowiedź, to przejmuje kontrolę, a rola węzła **W** kończy się.

- **Algorytm pierścieniowy**. W ramach tego algorytmu przyjmujemy, że węzły są fizycznie i logicznie tak uporządkowane, że każdy zna swojego następcę. Gdy jakikolwiek węzeł zauważy brak działania koordynatora, buduje komunikat ELEKCJA, który zawiera jego własny priorytet, i nadaje go do swojego następcy. Jeśli jego następca jest wyłączony, to nadawca pomija go i kontaktuje się z następnym członkiem pierścienia lub kolejnym, występującym za nim - aż do odnalezienia działającego węzła. W każdym kroku nadawca dodaje swój priorytet do listy w komunikacie, zgłaszając w ten sposób swoją kandydaturę w wyborach koordynatora.

  Ostatecznie komunikat dochodzi do procesu, który zapoczątkował jego obieg. Węzeł ten rozpoznaje to zdarzenie po odbiorze komunikatu z własnym priorytetem. W tym miejscu następuje zmiana typu komunikatu na KOORDYNATOR i obieg się powtarza, tym razem po to, by zawiadomić wszystkie inne węzły o nowym koordynatorze (wątek, który na utworzonej liście ma najwyższy priorytet) oraz o członkach nowego pierścienia. Po jednokrotnym obiegu komunikat ten jest usuwany i wszystko powraca do pracy.

  Zaimplementowanie, któregokolwiek z wyżej wymienionych algorytmów gwarantuje, że system będzie działał tak długo jak będzie przynajmniej jeden działający węzeł. Oczywistym następstwem tego, jest zwiększona niezawodność systemu.

B Testy wydajności bazy danych MySQL - dostępne narzędzia

B.1 Pomiary wydajności w systemach bazodanowych

Najczęściej stosowaną metodą pomiaru wydajności według [MyS05] jest zmierzenie ilości transakcji, jaką aplikacja może wykonać na bazie danych w ciągu sekundy (TPS) (ang. *Transactions Per Second*). W takim wypadku transakcją będziemy nazywali pojedyncze odwołanie aplikacji do bazy danych. Może to być prosta kwerenda pobierająca dane lub złożone zapytanie wykorzystujące procedury składowane w bazie danych. W takim kontekscie "transakcjaśnie koniecznie odnosi sie do ogólnie rozumianych transakcji typu ACID (ang. *Atomicity, Consistency, Isolation, Durability*), dostępnych w bazach danych, choć może takie transakcje zawierać. Wszystko zależy od konstrukcji zapytań testowych.

Wartość pomiarów TPS jest ściśle powiązana z typem transakcji, jakie zostały użyte podaczas testów. Użycie prostych zapytań odczytu podczas testu da nam zupełnie inny wynik niż test polegający na wstawieniu danych do wielu tabel, dlatego ogólna wartość TPS nie jest zbyt przydatna. Wymierne wyniki uzyskamy porównując metryki TPS z wielu pomiarów różniących się niewielką ilością parametrów. Wyniki uzyskane z pomiarów TPS w kompletnie różniących się testach są bezużyteczne, zatem nie ma sensu ich porównywanie.

B.1.1 Parametry wpływające na wydajność

Poniżej znajduje się lista parametrów systemu bazodanowego mających wpływ na ogólną wydajność bazy. Poprzez modyfikację tych czynników mamy realny wpływ na wydajność i możemy optymalizować działanie bazy danych.

Sprzęt Prędkość oraz architektura procesora, ilość jednostek CPU, prędkość szyny systemowej, prędkość dostępu do pamięci RAM, rozmiar pamięci ram, prędkość dostępu do dysków twardych, sprzęt sieciowy oraz prędkość interfejsów sieciowych - wszystkie te elementy są wykorzystywane przez bazę danych i mają znaczący wpływ na jej wydajność.

System Operacyjny Na wydajność bazy danych ma również wpływ wiele czynników związanych z systemem operacyjnym. System plików, zarządzanie pamięcią, obsługa i zarządzanie wielowątkowością, blokowanie dostępu do współdzielonych zasobów oraz organizacja schedulera to niektóre z aspektów, na które należy zwrócić uwagę przy testowaniu wydajności.

Ilość połączeń klienckich Liczba klientów próbujących jednocześnie skorzystać z zasobów bazy danych stanowi dobry wyznacznik wydajności systemu bazodanowego. Obsługa zwiększającej się ilości jednoczesnych połączeń stanowi duże wyzwanie dla bazy danych i znacząco wpływa na spadek wydajności systemu.

Poziom współbieżności bazy danych (ang. *Level of DB Concurrency*) Bezpośredni wpływ na równoległość obsługi żądań w bazie MySQL ma ilość wątków działających w bazie danych. W bazie MySQL jest to jednoznaczne z ilością nawiązanych połączeń. Zbyt mała ilość wątków

obsługujących nadchodzące połączenia spowoduje ograniczenie wydajności bazy, pomimo dostępnych zasobów sprzętowych. Natomiast zbyt duża ilość takich wątków może doprowadzić do nadmiernych kosztów związanych z obsługą i zarządzaniem wątkami, czego rezultatem będzie ograniczenie mocy potrzebnej na obsługę właściwych transakcji w bazie danych.

Data schema Struktura bazy danych ma istotny wpływ na wydajność. Zapytania realizowane na bazie zawierającej pojedynczą tabelę będą realizowane znacznie szybciej niż na bazie złożonej z 1000 tabel powiązanych ze sobą złożoną siecią kluczy obcych. W większości niezależnych testów logiczny schemat danych jest definiowany razem ze zbiorem transakcji, które będą wykonywane.

Ilość danych W zależności od rozmiaru bazy danych, wydajność będzie zależała od efektywności indeksowania danych w bazie. Im więcej danych przechowywanych jest w bazie, tym większy wpływ ma efektywna implementacja indeksowania.

Typ aplikacji Typowo aplikacje można podzielić na: read-only, read-mostly, read-write. W zależności od typu dostępu do danych w bazie, powinny zostać przeprowadzone różne typy testów. Dzięki temu będzie można stwierdzić jakie są zależności w wydajności między odczytem a zapisem danych do bazy. TPS jest dobrą miarą wykorzystywaną przy ocenie wydajności systemów OLTP.

Schemat dostępu do danych Większość aplikacji przez 90% czasu działania korzysta z mniej niż 5% danych zgromadzonych w bazie. Z tego powodu, jeśli chcemy testować wydajność samej bazy danych, a nie wydajność systemowego cache'u, powinniśmy przeprowadzić test czytając różne dane bezpośrednio z partycji dysku. Organizacja schematu dostępu do danych jest ważnym aspektem przy definiowaniu zbioru testów.

Konfiguracja bazy danych MySQL pozwala na modyfikację wielu parametrów wpływających na wydajność i pracę bazy. Należy odpowiednio zoptymalizować między innymi ustawienia maksymalnej ilości obsługiwanych równocześnie połączeń, rozmiar pamięci cache dla zapytań, styl logowania, rozmiar pamięci cache dla indeksów, protokół sieciowy przez który zaoferujemy dostęp do bazy oraz wiele innych parametrów, z których każdy może mieć wpływ na ogólną wydajność.

B.1.2 Jak testować?

Jak widać powyżej, liczba parametrów mająca wpływ na wyniki testów jest prawie nieograniczona i osiągnięcie optymalnej konfiguracji bazy danych nie jest zadaniem prostym. Najlepszym podejściem do przeprowadzenia testów wydajnościowych wydaje się być przeprowadzenie wielu prób zmieniając małą liczbę parametrów w każdym teście. Do testów można użyć jednego z wielu narzędzi testujących opisanych w kolejnych rozdziałach tego opracowania, kierując się następującymi wskazówkami:

- absolutna wartość TPS osiągnięta w różnych testach może być uznana za punkt odniesienia przy planowaniu i szacowaniu rzeczywistej wydajności bazy w środowisku produkcyjnym przy podobnej konfiguracji bazy.

- zmiany wartości TPS w poszczególnych testach pokażą wpływ pojedynczych parametrów systemu na ogólną wydajność

B.2 Narzędzia testujące (ang. *Benchmarking Tools*)

Dostępnych jest wiele narzędzi pozwalających na przeprowadzenie testów wydajności bazy danych MySQL. Bez potrzeby pisania własnych aplikacji, narzędzia te dostarczają gotowych danych, skryptów testowych oraz możliwość konfigurowania parametrów środowiska testowego, które dostarczy nam informacji o wydajności naszego środowiska produkcyjnego. Poniżej zostały zaprezentowane niektóre z dostępnych narzędzi.

B.2.1 SysBench

SysBench [Sys] jest modularnym niezależnym od platformy oraz wielowątkowym narzędziem testującym wydajność parametrów systemu operacyjnego mających wpływ na działanie bazy danych przy dużych obciążeniach. Narzędzie zostało napisane z myślą o przeprowadzaniu testów na bazie MySQL ale w przyszłości ma obsługiwać także inne implementacje baz danych. Głównym założeniem tego projektu jest szybka analiza wydajności systemu bez wcześniejszej skomplikowanej konfiguracji testów a nawet instalacji samej bazy.

Obecna wersja SysBench 0.3.1 pozwala na testowanie następujących parametrów:

- wydajność operacji I/O na plikach
- wydajność schedulera
- alokacja pamięci i prędkość transferu
- wydajność implementacji wątków POSIX
- wydajność serwera bazy danych (OLTP benchmark)

Istota działania SysBench-a jest bardzo prosta. Narzędzie uruchamia podaną przez nas liczbę wątków, które współbieżnie wykonują zapytania. Obciążenie generowane przez SysBench zależy od wybranego testu Można ograniczać całkowitą liczbę generowanych zapytań, czas wykonywania testu, bądź jedno i drugie. Dostępne testy są zaimplementowane w postaci wkompilowywanych modułów, a SysBench został zaprojektowany w taki sposób, aby dodawanie nowych modułów było proste.

Instalacja Proces instalacji i opis działania narzędzia są szczegółowo opisane w dokumentacji [Sys].
Instalacja polega pobraniu paczki z [Sys], rozpakowaniu jej i na wykoaniu następujących poleceń:

```
./configure
    make
    make install
```

Dodatkowo, jeżeli pliki nagłówkowe i biblioteki MySQL nie znajdują się w standardowych katalogach, należy je podać za pomocą opcji `--with-mysql-includes` oraz `--with-mysql-libs` w skrypcie `./configure`.

Użycie Podstawowe wywołanie wygląda następująco:

```
sysbench [common-options] --test=name [test-options]
```

Dostępne komendy to:

- `prepare`- wykonuje niezbędne przygotowania dla testów, które tego wymagają np. tworzy niezbędne pliki na dysku dla testu fileio
- `run`- wykonuje właściwy test wybrany przez opcję `--test=name`
- `cleanup`- czyści dane tymczasowe tworzone przez testy
- `help`- pokazuje informacje na temat użycia danego testu wybranego przez opcję `--test=name`

Dodatkowe opcje podawane w linii poleceń wraz z ustawieniami domyślnymi to:

```
--num-threads          1
--max-requests         10000
--max-time             0
--thread-stack-size    32K
--init-rnd             off
--test                 Required
--debug                off
--validate             off
--help                 off
--verbosity            4
--percentile           95
--batch                off
--batch-delay          300
--validate             off
```

Pełen opis parametrów dostępny jest w dokumentacji [Sys]

Test cpu Test polega na zliczaniu liczb pierwszych podczas wykonywania zapytania przez poszczególne działające współbieżnie wątki.
przykład uruchomienia:

```
sysbench --test=cpu --cpu-max-prime=20000 run
```

Test threads Testuje wydajność schedulera przy dużej liczbie wątków rywalizujących o dostęp do współdzielonych zasobów.
przykład uruchomienia:

```
sysbench --num-threads=64 --test=threads --thread-yields=100
         --thread-locks=2 run
```

Test mutex Testuje wydajność implementacji semaforów w momencie gdy wiele wątków działających współbieżnie blokuje semafor na krótką chwilę, po czym go odblokowuje.
dostępne opcje:

```
--mutex-num     4096
--mutex-locks   50000
--mutex-loops   10000
```

Test memory Testuje sekwencyjne odczyty i zapisy do pamięci. W zależności od opcji wątki odwołują się do lokalnego lub globalnego bloku pamięci.
dostępne opcje:

```
--memory-block-size  1K
--memory-scope       global
--memory-total-size  100G
--memory-oper        100G
```

Test fileio Test służy do zbadania obciążenia operacji I/O na plikach. W fazie prepare tworzone są pliki testowe, a następnie w run każdy z wątków wykonuje operacje I/O na tym pliku.
wspierane operacje I/O:

```
seqwr    sequential write
seqrewr  sequential rewrite
seqrd    sequential read
rndrd    random read
rndwr    random write
rndrw    combined random read/write
```

dostępne opcje:

```
--file-num              128
--file-block-size       16K
--file-total-size       2G
--file-test-mode        required
--file-io-mode          sync
--file-async-backlog    128
--file-extra-flags
--file-fsync-freq       100
--file-fsync-all        no
--file-fsync-end        yes
--file-fsync-mode       fsync
--file-merged-requests  0
--file-rw-ratio r       1.5
```

przykład uruchomienia:

```
$ sysbench --num-threads=16 --test=fileio --file-total-size=3G
           --file-test-mode=rndrw prepare
$ sysbench --num-threads=16 --test=fileio --file-total-size=3G
           --file-test-mode=rndrw run
$ sysbench --num-threads=16 --test=fileio --file-total-size=3G
           --file-test-mode=rndrw cleanup
```

Test oltp Ten test służy do przeprowadzanie rzeczywistych testów na bazie danych. W fazie prepare tworzona jest następująca testowa tabala w bazie danych:

```
CREATE TABLE `sbtest` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `k` int(10) unsigned NOT NULL default '0',
  `c` char(120) NOT NULL default '',
  `pad` char(60) NOT NULL default '',
  PRIMARY KEY  (`id`),
  KEY `k` (`k`);
```

Uruchomienie testu w trybie run pozwala na przeprowadzenie zarówno prostych jak i zaawansowanych transakcyjnych testów odczytu i zapisu.

Tryb SIMPLE pozwala na uruchomienie prostego zapytania przez każdy z działających wątków:

```
SELECT c FROM sbtest WHERE id=N
```

Tryb ADVANCED TRANSACTIONAL pozwala na testowanie zapytań transakcyjnych i w zależności od podanych opcji każda transakcja może składać się z:

- Point queries:

  ```
  SELECT c FROM sbtest WHERE id=N
  ```

- Range queries:

  ```
  SELECT c FROM sbtest WHERE id BETWEEN N AND M
  ```

- Range SUM() queries:

  ```
  SELECT SUM(c) FROM sbtest WHERE id BETWEEN N and M
  ```

- Range ORDER BY queries:

  ```
  SELECT c FROM sbtest WHERE id between N and M ORDER BY c
  ```

- Range DISTINCT queries:

```
SELECT DISTINCT c FROM sbtest WHERE id BETWEEN N and M ORDER BY c
```

- UPDATEs on index column:

```
UPDATE sbtest SET k=k+1 WHERE id=N
```

- UPDATEs on non-index column:

```
UPDATE sbtest SET c=N WHERE id=M
```

- DELETE queries:

```
DELETE FROM sbtest WHERE id=N
```

- INSERT queries:

```
INSERT INTO sbtest VALUES (...)
```

Tryb NON-TRANSACTIONAL jest podobny do trybu SIMPLE z tym, że można wybrać zapytanie, które ma być wykonywane. Dostępne są następujące zapytania:

- Point queries:

```
SELECT pad FROM sbtest WHERE id=N
```

- UPDATEs on index column:

```
UPDATE sbtest SET k=k+1 WHERE id=N
```

- UPDATEs on non-index column:

```
UPDATE sbtest SET c=N WHERE id=M
```

- DELETE queries:

```
DELETE FROM sbtest WHERE id=N
```

- INSERT queries:

```
INSERT INTO sbtest (k, c, pad) VALUES(N, M, S)
```

```
Dostępne opcje dla trybu testowania bazy danych to:
--oltp-test-mode          complex
--oltp-read-only          off
--oltp-range-size         100
--oltp-point-selects      10
--oltp-simple-ranges      1
--oltp-sum-ranges         1
--oltp-order-ranges       1
--oltp-distinct-ranges    1
--oltp-index-updates      1
--oltp-non-index-updates  1
--oltp-nontrx-mode        select
--oltp-connect-delay      10000
--oltp-user-delay-min     0
--oltp-user-delay-max     0
--oltp-table-name         sbtest
--oltp-table-size         10000
--oltp-dist-type          special
--oltp-dist-pct           1
--oltp-dist-res           75
--db-ps-mode              auto
```

Opcje do konfiguracji połączenia i ustawień bazy MySQL:

```
--mysql-host              localhost
--mysql-port              3306
--mysql-socket
--mysql-user              user
--mysql-password          password
--mysql-db                sbtest
--mysql-table-type        innodb
--myisam-max-rows         1000000
```

przykład uruchomienia:

```
$ sysbench --test=oltp --mysql-table-type=myisam
           --oltp-table-size=1000000
           --mysql-socket=/tmp/mysql.sock prepare
$ sysbench --num-threads=16 --max-requests=100000 --test=oltp
           --oltp-table-size=1000000 --mysql-socket=/tmp/mysql.sock
           --oltp-read-only run
```

B.2.2 MySQL Benchmark Suite

Narzędzie benchmark suite [MySb] jest udostępniane w źródłowych dystrybucjach MySQL-a, które można pobrać ze strony http://dev.mysql.com/downloads/. Narzędzie ma na celu spraw-

dzenie i powiadomienie użytkownika o wydajności operacji wykonywanych na konkretnej implementacji SQL. Narzędzie to działa w trybie jednowątkowym, więc mierzy minimalny czas wykonywania operacji. W przyszłości planowane jest dodanie wielowątkowości do narzędzia.

Użycie Po ściągnięciu i rozpakowaniu kodu źródłowego MySQL-a pakiet testujący wydajność powinien znajdować się w katalogu `sql-bench`. Aby uruchomić testy wydajnościowe należy skompilować MySQL-a i uruchomić skrypt `run-all-tests` z poziomu katalogu `sql-bench`.

```
shell> cd sql-bench
shell> perl run-all-tests --server=server_name
shell> perl run-all-tests --help
```

W pakiecie znajduje się również skrypt `crash-me` umożliwiający sprawdzenie możliwości i wspieranych przez system bazodanowy funkcji.

B.2.3 Open Source Database Benchmark

OSDB [OSD] jest narzędziem open-source zbudowanym na podstawie AS3AP (ANSI SQL Standard and Portable Benchmark). z myślą o projektantach baz danych i systemów bazodanowych. OSDB dostarcza gotowego kodu, który może być dowolnie modyfikowany i dostosowywany do potrzeb testowania wydajności baz danych. Szczegółowy opis działania, instalacji oraz użycia pakietu znajduje się w katalogu /docs w pakiecie instalacyjnym.

Instalacja Należy pobrać i rozpakować 2 pliki ze strony projektu: [OSD]

- `osdb-vx_y_z.tgz - (~60kB)` - kod źródłowy OSDB
- `osdb-data-4mb.tgz - (~2.7MB)` - dane do testów i rozwoju
- `osdb-data-40mb.tgz - (~27MB)` - dane do testów wydajnościowych

Następnie należy skompilować pobrany kod:

```
./configure --prefix=`pwd` --with-mysql=/usr/local/mysql
                           --with-postgresql=/usr/local/pgsql
make
make install
```

Uruchomienie testu OSDB przeprowadza testy wydajnościowe bazy danych i mierzy wydajność w kilku dziedzinach. Wspierane bazy danych to PostgreSQL, MySQL oraz Informix.

```
osdb [OPTION]
```

Opcje programu osdb są następujące:

```
--datadir path - specifies the directory name with
                 the data files
--short - omits the 15 minute warning time during
          multi-user sequence
--watch - allows for execution monitoring
          (until now same as watchall)
--watchall - allows for execution monitoring
--logfile file - specifies the log file name
--noindexes - omits the creation of indexes,
              primary keys and foreign keys
--nocreate - omits the DB structure creation sequence
--nomulti - doesn't perform the multi-user benchmark
--nosingle - doesn't perform the single-user benchmark
--users n - specifies the number of users for the
            multi-user benchmark
--help - dispalys help screen
--usage - shows informations about OSDB's usage
```

Dodatkowe opcje dla testów z bazą MySQL:

```
--mysql=innodb - Use InnoDB-style tables
--mysql=bdb - Use BDB-style tables
```

Dodatkowe opcje dla testów z bazą PostgreSQL

```
--postgresql=no_hash_index - Use BTree indexes instead
                             of hash tables
```

Przykłady użycia Kilka przykładów użycia OSDB wraz z krótkimi komentarzami:

```
./osdb-my -generate-files -size 40m -datadir /tmp
```

Tworzymy dane do testów.

```
./osdb-my -mysql=innodb
```

Wykonujemy standardowe testy z domyślnymi ustawieniami.

```
./osdb-my-ui --nocreate --nosingle --watchall
```

Ten przykład pokazuje jak uruchomić OSDB z baza MySQL przez interfejs użytkownika mysql aby obserwować jedynie test dla wielu użytkowników

```
./osdb-my --logfile osdb-my.log
```

Uruchamiamy OSDB na bazie MySQL ze standardowym API. Używamy pliku .log do przeanalizowania wykonania testu oraz błędów. Plik z logiem jest tworzony w bieżącym katalogu.

B.2.4 BenchW

BenchW [Ben] jest narzędziem stworzonym do porównywania różnych managerów baz danych na potrzeby magazynów danych (ang. *data warehaouses*). BenchW został zaprojektowany do testowania ładowania danych, tworzenia indeksów i badania wydajności zapytań w bazach danych. Głównym założeniem tego narzędzia jest wypełnienie luki na rynku pomiędzy złożonymi i skomplikowanymi narzędziami komercyjnymi a domorosłymi projektami. BenchW ma być prosty ale zapewniać na tyle rzeczywistych cech aby być narzędziem przydatnym w testowaniu wydajności baz danych.

Instalacja Instalacja pakietu została przygotowana przy użyciu Autoconf-a i przebiega standardowo:

```
./configure
make
make install
```

Kolejnym krokiem jest wygenerowanie danych testowych oraz skryptów. Należy do tego celu użyć binariów stworzonych podczas kompilacji. Należy wykonać następujące kroki:

- wygenerować dane przy pomocy programu loadgen
- wygenerować schemat bazy oraz skrypty z indeksami przy pomocy programu schemagen
- wygenerować skrypty ładujące przy pomocy programu loadgen
- wygenerować skrypty z zapytaniami przy pomocy programu querygen

Szczegółowe opcje dla poszczególnych programów wymienionych powyżej są opisane w pliku INSTALL w katalogu ze źródłami BenchW.

Ostatnim etapem przed uruchomieniem testów jest przygotowanie bazy danych do testów. Należy utworzyć testową bazę i uruchomić przygotowany wcześniej skrypt schema.sql.

Uruchomienie testu Aby przeprowadzić testy należy wczytać wygenerowane dane testowe do bazy (load.sql), utworzyć indeksy w bazie (indexes.sql) a na końcu uruchomić zapytania (qtype0-4.sql). Odnotowany czas wykonania zapytań będzie wynikiem wydajności naszej bazy danych. Test należy powtórzyć kilkakrotnie z różnymi ustawieniami bazy, indeksów oraz innych parametrów wpływających na wydajność bazy, które zostały opisane na początku opracowania. Porównanie czasów wykonania tych samych zapytań przy różnych ustawieniach da nam optymalne ustawienie parametrów pracy bazy danych.

B.2.5 Open Source Development Labs' (OSDL) Database Testing Suite (DBT)

OSDL opracowało zestaw profesjonalnych testów obciążeniowych [ODT] oraz testowy framework do przeprowadzania tych testów. Testy zostały zaprojektowane z myślą o profesjonalnych zastosowaniach Linuxa w środowiskach produkcyjnych. Dostępne są specjalnie przygotowane testy dla 4 różnych modeli obciążenia systemów bazodanowych:

- OSDL Database Test 1 (DBT-1)- testuje wydajność transakcyjnych systemów sieciowych. Symulowana jest aktywność użytkowników przeglądających strony www i kupujących produkty w sklepach on-line. Wyniki tego testu zawierają między innymi ilość transakcji obsługiwanych na sekundę (TPS), obciążenie procesora, aktywność operacji I/O oraz zużycie pamięci.
- OSDL Database Test 2 (DBT-2)- jest testem dla systemów OLTP. Symuluje dostęp kilku pracowników do bazy hurtowni produktów zmieniających dane klientów oraz sprawdzających informacje o produktach przechowywanych w bazie Wyniki tego testu zawierają między innymi ilość transakcji obsługiwanych na sekundę (TPS), obciążenie procesora, aktywność operacji I/O oraz zużycie pamięci.
- OSDL Database Test 3 (DBT-3)- symuluje obciążenie systemu wspierającego decyzje. Zawiera zestaw zapytań zorientowanych na podejmowanie decyzji biznesowych na podstawie danych w bazie oraz współbieżnie modyfikuje dane w bazie.
- OSDL Database Test 4 (DBT-4)- symuluje obciążenie serwera aplikacyjnego udostępniającego usługi sieciowe (ang. *Web services*). Test składa się z zadań podobnych jakie występują w transakcyjnym środowisku business-to-business.

Instalacja Przedstawiony zostanie przykładowy proces instalacji dla DBT-2. Należy zbudować binaria z kodu źródłowego dostępnego na stronie projektu zgodnie z opisem w pliku README-MYSQL.

```
./configure --with-mysql=DIR --with-mysql-includes=INC_PATH
            --with-mysql-libs=LIB_PATH
make
make install
```

Następnie należy wygenerować pliki testowe:

```
datagen -w 3 -d /tmp/dbt2-w3-mysql
```

Oraz załadować dane do bazy:

```
cd scripts/mysql
./mysql_load_db.sh

usage: mysql_load_db.sh [options]
options:
 -d <database name>
 -f <path to dataset files>
 -m <database scheme [OPTIMIZED|ORIG] (default scheme OPTIMIZED)>
 -c <path to mysql client binary. (default /usr/bin/mysql)>
 -s <database socket>
 -h <database host>
 -u <database user>
```

```
 -p <database password>
 -e <storage engine: [MYISAM|INNODB|BDB]. (default INNODB)>
 -l <to use LOCAL keyword while loading dataset>
 -v <verbose output>

Example:
sh mysql_load_db.sh -d dbt2 -f /tmp/dbt2-w3 -s /tmp/mysql.sock
```

Uruchomienie testu

```
cd scripts
sh ./run_mysql.sh

usage: run_mysql.sh -c <number of database connections> -t
       <duration of test> -w <number of warehouses>

other options:
 -n <database name. (default dbt2)>
 -h <database host name. (default localhost)>
 -l <database port number. (default 3306)>
 -o <database socket>
 -u <database user>
 -p <database password>
 -s <delay of starting of new thread in milliseconds(default 300ms)>
 -k <stack size. (default 256k)>
 -m <terminals per warehouse. [1..10] (default 10)>
 -z <comments for the test>
 -e <use zero delays for test (default no)>
 -v <verbose output>

Example: sh run_mysql.sh -c 20 -t 300 -w 3
```

Test będzie trwał 300 sekund z 20 połączeniami do bazy danych i 3 magazynami danych.

Po wykonaniu testów wyniki zostaną zapisane do plików i mogą zostać przeanalizowane. Następujące pliki mogą okazać się pomocne:

```
scripts/output/<number>/client/error.log - errors from
                                           backend C|SP based
scripts/output/<number>/driver/error.log - errors from
                                           terminals(driver)
scripts/output/<number>/driver/mix.log - info about performed
                                         transactions
scripts/output/<number>/driver/results.out - results of the test
```

C Dokumentacja techniczna autorskiego rozwiązania klastrowego

C.1 Demon zarządzający

C.1.1 Język programowania

Do implementacji demona zarządzającego wybrany został język Perl. Dodatkowe skrypty wspomagające mogą zostać zaimplementowane w Shell'u. Wykorzystane zostaną standardowe polecenia systemu operacyjnego Linux oraz specyficzne funkcje środowiska OpenSSI.

C.1.2 Analiza funkcjonalna

Analiza funkcjonalna przedstawia opis funkcjonalności projektowanego rozwiązania programistycznego. Demon zarządzający powinien realizować następujące funkcje:

1. Czytanie pliku konfiguracyjnego.
   (a) Analiza poprawności składni.
   (b) Parsowanie treści w celu wyodrębnienia danych konfiguracyjnych węzłów MySQL.
   (c) Parsowanie treści w celu wyodrębnienia wartości zmiennych definiujących minimalną liczbę węzłów danego typu.
2. Tworzenie i modyfikacja plików konfiguracyjnych węzłów MySQL Cluster.
   (a) Generowanie na podstawie szablonu i aktualnych dynamicznych danych konfiguracyjnych skryptów węzłów danych w znanej lokalizacji.
   (b) Generowanie na podstawie szablonu i aktualnych dynamicznych danych konfiguracyjnych skryptów węzłów SQL w znanej lokalizacji.
   (c) Generowanie na podstawie szablonu i aktualnych dynamicznych danych konfiguracyjnych skryptów węzła zarządzającego w znanej lokalizacji.
3. Uruchomienie węzłów systemu MySQL Cluster
   (a) Sprawdzenie liczby dostępnych komputerów klastra w odniesieniu do danych zawartych w pliku konfiguracyjnym.
   (b) Weryfikacja poprawności nadanych adresów IP.
   (c) Uruchomienie standardowego zestawu usług w odpowiedniej kolejności według zadanej konfiguracji.
   (d) Zapamiętanie numerów *PID* wszystkich uruchomionych procesów w celu późniejszego monitorowania.
4. Monitorowanie procesów systemu MySQL Cluster i obsługa sytuacji awaryjnych.
   (a) Monitorowanie stanu procesów uruchomionych w klastrze o pamiętanych numerach *PID*.

(b) Identyfikacja sytuacji awaryjnej - zniknięcie procesu z listy procesów uruchomionych na klastrze.

(c) Obliczenie nowej liczby procesów danego typu i porównanie z wartością minimalną.

(d) Dynamiczna reakcja na awarię - przywrócenie minimalnej liczby działających węzłów danego typu.

i. Stworzenie listy dostepnych komputerów klastra.

ii. Porównanie listy dostępnych komputerów z informacjami o uruchomionych procesach.

iii. Podjęcie decyzji, na którym komputerze uruchomić nowy proces danego typu - wybrany zostaje pierwszy z wolnych, dostępnych komputerów.

iv. Zmiana plików konfiguracyjnych procesu.

v. Sprawdzenie czy na danej maszynie nie istnieje proces danego typu. Zabicie ewentualnie istniejącego procesu oraz uruchomienie nowego procesu.

vi. Zmiana plików konfiguracyjnych pozostałych procesów.

vii. Przeładowanie konfiguracji pozostałych procesów poprzez wysłanie sygnału SIGHUP.

5. Zakończenie wszystkich usług MySQL Cluster podczas kończenia pracy demona.

C.2 Składowanie danych na dysku

C.2.1 Skrypt konfiguracyjny SQL

Zakładając prawidłową konfigurację wszystkich węzłów systemu MySQL Cluster w wersji co najmniej 5.1.6, konfiguracja składowania tabel NDB w przestrzeni dyskowej może być zrealizowane w trzech opisanych poniżej krokach.

1. Stworzenie grupy i przypisanie plików logów *UNDO*.

2. Stworzenie przestrzeni tabel i przypisanie do niej plików danych.

3. Stworzenie tabel wykorzystujących stworzoną przestrzeń tabel do przechowywania danych.

Każdy z wymienionych kroków polega, ściśle rzecz biorąc, na wykonaniu zapytań SQL według opisanych w dokumentacji wytycznych.

1. Należy stworzyć grupę plików logów `lg_1` używając polecenia `CREATE LOGFILE GROUP`. Grupa ta składa się z dwóch plików logów *UNDO* o nazwach `undo_1.dat` i `undo_2.dat` o początkowych rozmiarach odpowiedno 16 MB i 12 MB. Opcjonalnie można podać rozmiar bufora grupy. Grupa plików logów musi być stworzona z conajmniej jednym zadanym plikiem. Prawidłowe zapytanie SQL powinno wyglądać następująco:

```
CREATE LOGFILE GROUP lg_1
    ADD UNDOFILE 'undo_1.dat'
    INITIAL_SIZE 16M
    UNDO_BUFFER_SIZE 2M
    ENGINE NDB;
```

Dodanie drugiego pliku logu *UNDO* następuje poprzez wykonanie zapytania SQL:

```
ALTER LOGFILE GROUP lg_1
    ADD UNDOFILE 'undo_2.dat'
    INITIAL_SIZE 12M
    ENGINE NDB;
```

Plik logu tworzony jest w katalogu wskazywanym przez zmienną `DataDirectory` podaną w plikach konfiguracyjnych.

2. Kolejnym etapem jest stworzenie przestrzeni tabel. Przestrzeń ta zawiera pliki danych, w których dane są składowane. Przestrzeń jest również powiązana z konkretną grupą plików logów. Poniższe zapytania realizują zadanie stworzenia przestrzeni tabel i przypisania odpowiednich plików.

```
CREATE TABLESPACE ts_1
    ADD DATAFILE 'data_1.dat'
    USE LOGFILE GROUP lg_1
    INITIAL_SIZE 32M
    ENGINE NDB;

ALTER TABLESPACE ts_1
    ADD DATAFILE 'data_2.dat'
    INITIAL_SIZE 48M
    ENGINE NDB;
```

3. Po wykonaniu poprzednich kroków można przystąpić do tworzenia tabel w nowo stworzonej przestrzeni tabel dyskowych. Klauzula `STORAGE DISK` nakazuje serwer MySQL składować dane w plikach na dysku twardym. Zapytanie stworzenia tabeli NDB składowanej w plikach dyskowych wygląda następująco:

```
CREATE TABLE dt_1 (
    member_id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
    last_name VARCHAR(50) NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    dob DATE NOT NULL,
    joined DATE NOT NULL,
    INDEX(last_name, first_name)
    )
    TABLESPACE ts_1 STORAGE DISK
    ENGINE NDB;
```

C.3 Rozszerzenie aplikacji klienckiej

C.3.1 Język programowania

Do implementacji demona zarządzającego wybrany został język Perl. Dodatkowe skrypty wspomagające mogą być zaimplementowane w Shell'u. Wykorzystane zostaną standardowe polecenia systemu operacyjnego Linux oraz specyficzne funkcje środowiska OpenSSI.

C.3.2 Analiza funkcjonalna

1. Wyszukiwanie dostępnych węzłów SQL.
   (a) Czytanie pliku konfiguracyjnego ze znanej lokalizacji.
   (b) Parsowanie pliku w celu wyodrębnienia informacji konfiguracyjnych.
   (c) Weryfikacja dostępności węzła SQL pod zadanym adresem.
2. Wybór optymalnego węzła SQL.
   (a) Badanie obciążenia dostępnych węzłów - na podstawie danych zwróconych przez polecenie `loads` .
   (b) Wybór najmniej obciążonego w danej chwili węzła.
3. Uruchomienie procesu klienta.
   (a) Modyfikacja pliku konfiguracyjnego klienta - podstawienie odpowiedniego adresu IP.
   (b) Uruchomienie procesu klienta standardowym poleceniem.

Literatura

[Atk03] Leon Atkinson. *Core MySQL. Przewodnik zaawansowanego programisty*. Wydawnictwo Helion, 2003.

[Ben] BenchW. http://benchw.sourceforge.net/.

[con] http://dev.mysql.com/doc/refman/5.0/en/mysql-cluster-config-file.html.

[EN05] Ramez Elmasri and Shamkant B. Navathe. *Wprowadzenie do systemów baz danych*. Wydawnictwo Helion, 2005.

[FN06] Krzysztof Fabjanski and Janusz Krzysztof Nowacki. *Dokumentacja techniczna paczki nr 1 silnika bazy danych MySQL*. 2 kwietnia 2006.

[Maz] Wojciech Maziarz. Realizacja niezawodnych usług w rozwiązaniu ssi. http://www.ia.pw.edu.pl/~tkruk/openssi/ha_wmaziarz.pdf.

[MS02] Neil Matthew and Richard Stones. *Zaawansowane programowanie w systemie Linux*. Wydawnictwo Helion, 2002.

[mysa] Strona domowa *MySQL*. http://www.mysql.com/.

[MySb] MySQL Benchmark Suite. http://dev.mysql.com/doc/mysql/en/mysql-benchmarks.html.

[MyS05] Mysql performance benchmarks - measuring mysql's scalability and throughput. Technical while paper, MySQL, http://corpo.mandriva.com/xwiki/bin/download/Main/Technology/mysql-performance-whitepaper.pdf, March 2005.

[Naj] Marcin Najs. Realizacja niezawodnych usług w openssi. http://www.ia.pw.edu.pl/~tkruk/openssi/ha_mnajs.pdf.

[ODT] Open Source Development Labs' (OSDL) Database Testing Suite (DBT.

[OSD] Open Source Database Benchmark. http://osdb.sourceforge.net/.

[Ost] Karol Ostrowski. Niezawodne usługi w rozwiązaniach ssi. http://www.ia.pw.edu.pl/~tkruk/openssi/ha_kostrowski.pdf.

[Pal06] Rafał Paluch. *Opis i analiza istniejacych rozwiazan w ramach MySQL umozliwiajacych realizacje klastrowosci, zwiekszenia niezawodnosci i wydajnosci serwera bazy danych*. 28 marca 2006.

[qui] http://dev.mysql.com/doc/refman/5.0/en/mysql-cluster-quick.html.

[Ron05] Mikael Ronstrom. Recovery principles of mysql cluster 5.1, 2005. http://www.vldb2005.org/program/paper/wed/p1108-ronstrom.pdf.

[Sys] SysBench Homepage. http://sysbench.sourceforge.net/.

[wik] *Wikipedia, wolna encyklopedia*. http://pl.wikipedia.org/wiki/MySQL.