****Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовой проект**

**по курсу «Операционные системы»**

**Распределенное хранилище данных**

Группа: М80 – 207Б – 18

Студент: Син Денис Дмитриевич

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2019.

**Содержание**

1. Постановка задачи
2. Общие сведения о программе
3. Общий метод и алгоритм решения
4. Демонстрация работы программы
5. Вывод

**Постановка задачи**.

Реализовать распределенное хранилище данных, которое представляет из себя два сервера, на которых хранятся данные, состоящие из пар ключ и значение, где ключ типа int, а значение типа double. И должна быть библиотека, которая реализует API с данным хранилищем данных, которое поддерживает 3 типа команд:

1. put key val
2. get key
3. remove key

В случае неисправности в работе одного из серверов данные команды все равно должны работать в нормальном режиме.

Данные между клиентом и серверами передаются с помощью технологии очереди сообщений

**Общие сведения о программе**

Проект состоит из трёх исполняемых файлов store1, store2, app. При исполнении файлов store1 config и store2 config запускаются два сервера, каждый из которых может сохранять, отдавать или удалять данные клиенту при различных запросах. А также при выполнении команды сервером store1, он реплицирует данные на сервер store2.

Для удобства работы были созданы классы MainStore и ReserveStore, которые наследуются от класса BaseStore. Реализация данных классов находится в файле storage.h.

Реализация API к хранилищу находится в файле storage\_api.h. В нем реализованы функции get, put, remove. В случае если все серверы мертвые при использовании данных функций бросается исключение.

Также для удобства использования библиотеки zmq.hpp были созданы класс IMessage – это абстрактный класс, от которого наследуются разные типы сообщений и namespace функций, которые занимаются отправкой и получением сообщений, оперируя с абстрактным классом IMessage.

**Общий метод и алгоритм решения**.

В процессе работы пользовательской программы с хранилищем данных возникают два сценария работы. Первый сценарий заключается в запросе пользовательской программы первого сервера, он отдает ответ обратно и реплицирует данные на резервный сервер. Второй сценарий заключается в следующем: если при первый сервер недоступен делается запрос на резервный сервер. В итоге получается, что клиентская часть должна знать адреса обоих серверов. А главный сервер должен знать адрес резервного.

Схема взаимодействия главного и резервного серверов построена на push/pull сокетах. Главный сервер просто перебрасывает команду от клиента на резервный сервер. Таким образом при падении одного из серверов хранилище будет работать корректно.

**Демонстрация работы программы.**

~/CSinDenis/university/sem3/OperatingSystems/KP/cmake-build-debug(master\*) » cat basic\_test

put 1 2

put 2 3

put 3 4

put 4 5

put 5 6

put 6 7

put 7 8

put 8 9

put 9 10

put 10 11

put 11 12

put 12 13

put 13 14

put 14 15

put 15 16

put 16 17

put 17 18

put 18 19

put 19 20

get 1

get 2

get 3

get 4

get 5

get 6

get 7

get 8

get 9

get 10

get 11

get 12

get 13

get 14

get 15

get 16

get 17

get 18

get 19

remove 13

remove 14

remove 15

remove 16

get 10

get 11

get 12

get 13

get 14

get 15

~/CSinDenis/university/sem3/OperatingSystems/KP/cmake-build-debug(master\*) » ./app < basic\_test

ok

ok

ok

ok

ok

ok

ok

ok

ok

ok

ok

ok

ok

ok

ok

ok

ok

ok

ok

ok: 2

ok: 3

ok: 4

ok: 5

ok: 6

ok: 7

ok: 8

ok: 9

ok: 10

ok: 11

ok: 12

ok: 13

ok: 14

ok: 15

ok: 16

ok: 17

ok: 18

ok: 19

ok: 20

ok

ok

ok

ok

ok: 11

ok: 12

ok: 13

not found

not found

not found

**Вывод**

В курсовом проекте было реализовано распределенное хранилище данных, которое расположено на двух серверах. По сравнению с лабораторной работой №6 курсовой проект был легче и к тому же появился опыт работы с zmq, поэтому получилось сосредоточиться именно на проектировании проекта. В отличии от лабораторной работы №6 теперь сообщение представлено не одним классом, а для каждого отдельного типа сообщения создан свой класс, где все данные классы являются наследниками абстрактного класса. Дополнительно для каждого класса сообщения были реализованы сереализации и десереализация. Таким образом довольно просто работать как с отправкой, так и с получением данных. Также в процессе выполнения данной работы появилось более ясное представление, как работают кэши типа redis и вообще различные сервисы.