ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»

Институт радиоэлектроники и информационных технологий
Кафедра “Прикладная математика”

Отчет по лабораторной работе курса “Базы данных”

Лабораторная работа №5
«NoSQL-Cassandra»

Выполнил студент группы 18-ПМ:
Винокуров М.С
Проверил:
Моисеев А.Е.

НИЖНИЙ НОВГОРОД

2021 г.

Оглавление

Введение 3
Cassandra 3
Задание 5
Выполнение работы 6
Вывод 12
Список источников 13
Приложение. Исходный код программы 14
GameDB-5.js 14
db5.html 18

Введение

Cassandra

Apache Cassandra — распределённая система управления базами данных, относящаяся к классу NoSQL-систем и рассчитанная на создание высокомасштабируемых и надёжных хранилищ огромных массивов данных, представленных в виде хэша.

Изначально проект был разработан в недрах Facebook и в 2009 году передан под крыло фонда Apache Software Foundation, эта организация продолжает развитие проекта. Промышленные решения на базе Cassandra развёрнуты для обеспечения сервисов таких компаний, как Cisco, IBM, Cloudkick, Reddit, Digg, Rackspace, Apple и Twitter. К 2011 году крупнейший кластер серверов, обслуживающий единую базу данных под управлением Cassandra, насчитывал более 400 машин и содержал данные размером более 300 ТБ.

Написана на языке Java, реализует распределённую hash-систему, сходную с DynamoDB, что обеспечивает практически линейную масштабируемость при увеличении объёма данных. Использует модель хранения данных на базе семейства столбцов (ColumnFamily), чем отличается от систем, подобных MemcacheDB, которые хранят данные только в связке «ключ — значение», возможностью организовать хранение хэшей с несколькими уровнями вложенности. Относится к категории отказоустойчивых СУБД: помещённые в базу данные автоматически реплицируются на несколько узлов распредёленной сети или даже равномерно распределяются в нескольких дата-центрах, при сбое узла его функции на лету подхватываются другими узлами, добавление новых узлов в кластер и обновление версии Cassandra производится на лету, без дополнительного ручного вмешательства и переконфигурации других узлов. Тем не менее настоятельно рекомендуется заново сгенерировать ключи (токены) для каждого узла, включая существующие, чтобы сохранить качество распределения нагрузки. Генерации ключей для существующих узлов можно избежать в случае кратного увеличения количества узлов (в 2 раза, в 3 раза и так далее).

Для упрощения взаимодействия с базой данных поддерживается язык формирования структурированных запросов CQL (Cassandra Query Language), который в какой-то степени сходен с SQL, но существенно урезан по функциональным возможностям. Например, можно выполнять только простейшие запросы SELECT с выборкой по определённому условию. Добавление и обновление осуществляется через единое выражение UPDATE, операция INSERT отсутствует (если записи нет, при выполнении UPDATE она создаётся — используется семантика SQL-оператора MERGE). Из отличительных возможностей — поддержка пространств имён и семейств столбцов, создание индексов через выражение «CREATE INDEX». Драйверы с поддержкой CQL реализованы для языков Python (DBAPI2), Java (JDBC), Ruby (gem cassandra-cql), PHP (Thrift, cassandra-pdo, Cassandra-PHP-Client-Library), JavaScript (Node.js) и Perl (DBD::Cassandra).

Кроме того, CQL реализован в СУБД Scylla[en], которая архитектурно и лингвистически повторяет систему Cassandra, но написана на C++ с целью повышения показателей производительности.

Задание

- Создать базу данных Cassandra из 20-ти объектов
- Вывести на веб-страницу содержимое базы данных:
    - На странице кроме таблицы с содержимом базы данных находятся поля для ввода фильтров
    - Содержимое полей отправляется на сервер
    - Сервер на основе содержимого фильтров формирует запрос к базе данных
    - Общение между клиентом и серверов в формате JSON
    - Таблица формируется на стороне клиента

Выполнение работы

Был создан проект Node.JS.
В качестве объектов хранящихся в базе данных выступают «игры».

В начале программы подключаются необходимые модули

```
var cassandra = require("cassandra-driver");
var fs = require("fs")
var http = require("http");
```

Выполняется соединение с базой данных

```
var client = new cassandra.Client({ contactPoints: ['127.0.0.1'], keyspace: "syst
em", localDataCenter: "datacenter1" });
```

Делается обращение к базе данных, для проверки наличия таблицы

```
client.execute("SELECT * FROM system_schema.keyspaces WHERE keyspace_name='games'
;", function (err, result)
```

В случае её отсутствия выполняется создание и заполнение таблицы

```
client.execute("USE games;");
                client.execute("CREATE TABLE GameDB (id double PRIMARY KEY, game
TEXT, platform TEXT, price double, release_date double);", function (_err, _res)
{
                    if (err) {
                        console.error(err);
                        return;
                    }
                    //fill here
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(1, ?, ?, ?, ?);", ["Fallout", "Bethesda Laucnher", 35, 199
4]);
```

Частичный код заполнения

После этого выполняется отправка сообщения о готовности к созданию сервера

```
process.emit("readyToServerCreate");
```

По которому выполняется создание и вывод сообщения в консоль

```
process.on('readyToServerCreate', () => {
    http.createServer(server_callback).listen(3000);
    console.log("Listen at http://localhost:3000/");
```

В функции обработки запроса к серверу

```
var server_callback = function (request, response) {
    console.log("request to: " + request.url + " method: " + request.method)
    if (request.method == "GET") {
        handle_GET(request, response);
    } else {
        handle_POST(request, response);    }}
```

Разделяются POST и GET запросы к серверу
В обработчике GET запросов

```
var handle_GET = function (request, response) {
    switch (request.url) {
        case "/":
            fs.readFile("./db5.html", function (err, content) {
                if (!err) {
                    response.writeHead(200, { "Content-
Type": "text/html; charset=utf-8" });
                    response.end(content, "utf-8")
                } else {
                    response.writeHead(500, { "Content-
Type": "text/plain; charset=utf-8" });
                    response.end(err.message, "utf-8");
                    console.log(err);
                }
            });
            break;
        default:
            response.writeHead(404, { "Content-Type": "text/html; charset=utf-
8" });
            response.end("<!DOCTYPE html>\n" +
                "<html>\n" +
                "   <head>\n" +
                "       <meta charset='utf-8'>\n" +
                "   </head>\n" +
                "   <body>\n" +
                "404, NOT FOUND: " + request.url +
                " \n</body>\n" +
                "</html>"
            );
    }
}
```

Клиенту отправляется либо содержимое файла db5.html, либо сообщение об ошибке.

В обработчике POST запросов

```
var handle_POST = function (request, response) {
    if (request.url != "/get_table") {
        response.writeHead(500, { "Content-Type": "text/plain; charset=utf-8" });
        response.end();
    }
    ///////.....///////
    var data = '';
    request.on('data', function (chunk) {
        data += chunk;
    });
```

формируется полученный от клиента объект в формате JSON и котором хранятся значения фильтров.

Выполняется запрос к базе данных (из-за особенностей Cassandra объединить запросы с возможно пустыми строками невозможно и приходится обрабатывать все ситуации вручную) с передачей выполнения в функцию

```
    var parse = function (err, res) {
        if (err) {
            console.log(err);
            response.writeHead(404, { "Content-Type": "text/plain; charset=utf-
8" });
            response.end();
        } else {
            response.writeHead(200, { "Content-
Type": "application/json; charset=utf-8" });
            var db_data = {};
            db_data.table = res.rows;
            response.end(JSON.stringify(db_data));
        }
    }
```

Которая анализирует результат запроса к базе и в случае успеха результат оправляется клиенту в формате JSON, в случае ошибки отправляет сообщение об ошибке.

Так же была создана веб-страница, в теле которой

```
<body>
    <p>
        GameDB from Cassandra.
    </p>
    <p>
       Game Search<input id="game" type="text"> Game Price from: <input id="from"
            type="number" value=0> to: <input id="to" type="number" value=100>$.
    </p>
    <p>
        <span id="res" style="font-style: italic"></span>
    </p>
</body>
```

Находятся 4 поля для ввода – фильтры и поле, в котором будет находится таблица с результатом запроса к серверу

При загрузке браузером страницы

```
window.onload = function () {
            var game_input = document.getElementById("game");
            game_input.oninput = function () {
                game_filter = game_input.value;
                update();
            };
            var fr_input = document.getElementById("from");
            fr_input.oninput = function () {
                price_filter_from = fr_input.value;
                update();
            };
            var to_input = document.getElementById("to");
            to_input.oninput = function () {
                price_filter_to = to_input.value;
                update();
            };
            update();
        }
```

Устанавливаются обработчики изменения значений в полях ввода – при их изменении вызывается функция отправки серверу новых значений и изменения содержимого таблицы

```
function update() {
            var filters = {};
            filters.game = game_filter;
            filters.price_from = price_filter_from;
            filters.price_to = price_filter_to;
            readServer("/get_table", JSON.stringify(filters), function (err, resp
onse) {
                if (err) document.getElementById("res").innerHTML = err;
                else {
                    var temp = "";
                    temp = "<table cellspacing=\"2\" border=\"1\" cellpadding=\"5
\">\n";
                    var rows = JSON.parse(response).table;
                    for (var i = 0; i < rows.length; i++) {
                        temp += "<tr><td>" + rows[i].game + "</td><td align=\"cen
ter\">" + rows[i].price + "$</td><td align=\"center\">" + rows[i].release_date +
"</td><td align=\"center\">" + rows[i].platform +"</td></tr>\n";
                    }
                    temp += "</table>";
                    document.getElementById("res").innerHTML = temp;
                }
            });
        }
```

В ней используется функция, отправляющая серверу фоновый запрос по технологии AJAX

```
readServer("/get_table", JSON.stringify(filters), function (err, response) {
                if (err) document.getElementById("res").innerHTML = err;
                else {
                    var temp = "";
                    temp = "<table cellspacing=\"2\" border=\"1\" cellpadding=\"5
\">\n";
                    var rows = JSON.parse(response).table;
                    for (var i = 0; i < rows.length; i++) {
                        temp += "<tr><td>" + rows[i].game + "</td><td align=\"cen
ter\">" + rows[i].price + "$</td><td align=\"center\">" + rows[i].release_date +
"</td><td align=\"center\">" + rows[i].platform +"</td></tr>\n";
                    }
                    temp += "</table>";
                    document.getElementById("res").innerHTML = temp;
                }
            }
```

Эта функция принимает на вход URL сервера(можно относительный), данные для отправки в теле запроса и функцию – обработчик результата.

После загрузки страницы:

HELLO CASSANDRA, I LOVE YOU

Game Search Game Price from: 0 to: 100 $.

Fallout	*35$*	*1994*	*Bethesda Launcher*
Fallout 2	*45$*	*1996*	*Bethesda Launcher*
Fallout Nevada	*0$*	*2003*	*non*
Fallout New Vegas	*40$*	*2009*	*Steam*
Halo Combat Evolved	*60$*	*2003*	*Xbox Live*
Halo 2	*60$*	*2006*	*Xbox Live*
Dota 2	*0$*	*2011*	*Steam*
League of legends	*0$*	*2009*	*RIOT Launcher*
Counter Strike - Condition Zero	*25$*	*2005*	*Steam*
Call of Duty 3	*60$*	*2006*	*Xbox Live*
Horizon Zero Dawn	*60$*	*2017*	*PS Network*
The last of Us	*60$*	*2013*	*PS Network*
Fallout 76	*50$*	*2018*	*Bethesda Launcher*
Halo 5	*60$*	*2015*	*Xbox Live*
Stubbs the Zombie	*45$*	*2004*	*Xbox Live*
Battlefield 1942	*55$*	*2003*	*Retail*
Team Fortress 2	*5$*	*2008*	*Steam*
Half-Life	*45$*	*1999*	*Retail*
Half-Life 2	*60$*	*2004*	*Steam*

После ввода фильтров

HELLO CASSANDRA, I LOVE YOU

Game Searchs Game Price from: 0 to: 3 $.

League of legends	*0$*	*2009*	*RIOT Launcher*

Вывод

В ходе выполнения работы был создан веб-сервер и веб-страница, которые взаимодействуют с помощью AJAX. Веб-сервер взаимодействует с базой данных и отправляет данные клиенту, на стороне клиента формируется таблица с содержимым базы данных и отправляются содержащиеся в полях ввода – фильтрах значения на сервер.

Список источников

1. Статья о Cassandra в Википедии - https://ru.wikipedia.org/wiki/Apache_Cassandra
2. Cassandra - http://cassandra.apache.org/

Приложение. Исходный код программы

GameDB-5.js

```
var cassandra = require("cassandra-driver");
var fs = require("fs")
var http = require("http");
var client = new cassandra.Client({ contactPoints: ['127.0.0.1'], keyspace: "syst
em", localDataCenter: "datacenter1" });

process.on('unhandledRejection', error => {
    // Will print "unhandledRejection err is not defined"
    console.log('unhandledRejection:', error.message);
});

client.execute("SELECT * FROM system_schema.keyspaces WHERE keyspace_name='games'
;", function (err, result) {
    if (err) {
        console.error(err);
        return;
    } else {
        if (result.rows.length != 0) {
            client.execute("USE games;");
            process.emit("readyToServerCreate");
            return;
        } else {
            client.execute("CREATE KEYSPACE games WITH REPLICATION={'class':'Simp
leStrategy', 'replication_factor':1};", function (err, _res) {
                if (err) {
                    console.error(err);
                    return;
                }
                client.execute("USE games;");
                client.execute("CREATE TABLE GameDB (id double PRIMARY KEY, game
TEXT, platform TEXT, price double, release_date double);", function (_err, _res)
{
                    if (err) {
                        console.error(err);
                        return;
                    }
                    //fill here
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(1, ?, ?, ?, ?);", ["Fallout", "Bethesda Laucnher", 35, 199
4]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(2, ?, ?, ?, ?);", ["Fallout 2", "Bethesda Laucnher", 45, 1
996]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(3, ?, ?, ?, ?);", ["Fallout Nevada", "non", 0, 2003]);
```

```
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(4, ?, ?, ?, ?);", ["Fallout NEw Vegas", "steam", 40, 2009]
);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(5, ?, ?, ?, ?);", ["Halo Combat Envolved", "Xbox Live", 60
, 2003]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(6, ?, ?, ?, ?);", ["Halo 2", "Xbox Live", 60, 2005]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(7, ?, ?, ?, ?);", ["Dota 2", "Steam", 0, 2011]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(8, ?, ?, ?, ?);", ["League of legends", "RIOT Launcher", 0
, 2009]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(9, ?, ?, ?, ?);", ["Counter Strike - Condition Zero", "Ste
am", 25, 2004]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(10, ?, ?, ?, ?);", ["Call of Duty", "Xbox Live", 45, 2005]
);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(11, ?, ?, ?, ?);", ["Horizon Zero Dawn", "PS Network", 60,
 2017]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(12, ?, ?, ?, ?);", ["The last of Us", "PS Netowrk", 60, 20
13]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(13, ?, ?, ?, ?);", ["Fallout 76", "Bethesda Laucnher", 50,
 2018]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(14, ?, ?, ?, ?);", ["Halo 5", "Xbox Live", 60, 2015]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(15, ?, ?, ?, ?);", ["Stubbs the Zombie", "Xbox Live", 45,
2004]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(16, ?, ?, ?, ?);", ["Battlefield 1942", "Retail", 55, 2002
]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(17, ?, ?, ?, ?);", ["Team Fortress 2", "Steam", 5, 2008]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(18, ?, ?, ?, ?);", ["Half-Life", "Steam", 45, 1999]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(19, ?, ?, ?, ?);", ["Half-Life 2", "Steam", 60, 2004]);
                    client.execute("INSERT INTO GameDB (id, game, platform, price
, release_date) VALUES(20, ?, ?, ?, ?);", ["Half-
Life 3", "CyberPunkedSteam", 120, 2077]);
                    client.execute("CREATE CUSTOM INDEX ON GameDB (game) USING 'o
rg.apache.cassandra.index.sasi.SASIIndex' WITH OPTIONS = {'mode': 'CONTAINS', 'an
alyzer_class': 'org.apache.cassandra.index.sasi.analyzer.StandardAnalyzer', 'case
_sensitive': 'false'};");
                    client.execute("CREATE CUSTOM INDEX ON GameDB (platform) USIN
G 'org.apache.cassandra.index.sasi.SASIIndex' WITH OPTIONS = {'mode': 'CONTAINS',
```

```
 'analyzer_class': 'org.apache.cassandra.index.sasi.analyzer.StandardAnalyzer', '
case_sensitive': 'false'};");
                    client.execute("CREATE INDEX ON GameDB (price);");
                    console.log("table GameDB created");
                    process.emit("readyToServerCreate");
                });
            });
        }
    }
});

process.on('readyToServerCreate', () => {
    http.createServer(server_callback).listen(3000);
    console.log("Listen at http://localhost:3000/");
});

var handle_GET = function (request, response) {
    switch (request.url) {
        case "/":
            fs.readFile("./db5.html", function (err, content) {
                if (!err) {
                    response.writeHead(200, { "Content-
Type": "text/html; charset=utf-8" });
                    response.end(content, "utf-8")
                } else {
                    response.writeHead(500, { "Content-
Type": "text/plain; charset=utf-8" });
                    response.end(err.message, "utf-8");
                    console.log(err);
                }
            });
            break;
        default:
            response.writeHead(404, { "Content-Type": "text/html; charset=utf-
8" });
            response.end("<!DOCTYPE html>\n" +
                "<html>\n" +
                "    <head>\n" +
                "        <meta charset='utf-8'>\n" +
                "    </head>\n" +
                "    <body>\n" +
                "404, NOT FOUND: " + request.url +
                " \n</body>\n" +
                "</html>"
            );
    }
}

var handle_POST = function (request, response) {
    if (request.url != "/get_table") {
        response.writeHead(500, { "Content-Type": "text/plain; charset=utf-8" });
```

```
        response.end();
    }
    var parse = function (err, res) {
        if (err) {
            console.log(err);
            response.writeHead(404, { "Content-Type": "text/plain; charset=utf-
8" });
            response.end();
        } else {
            response.writeHead(200, { "Content-
Type": "application/json; charset=utf-8" });
            var db_data = {};
            db_data.table = res.rows;
            response.end(JSON.stringify(db_data));
        }
    }

    var data = '';
    request.on('data', function (chunk) {
        data += chunk;
    });
    request.on('end', function () {
        var filters = JSON.parse(data);
        if (filters.game != "")
            client.execute("SELECT * FROM GameDB WHERE " +
                "game LIKE ? AND " +
                "price>=? AND price<=? ALLOW FILTERING;", ['%'+filters.game+'%',
parseInt(filters.price_from), parseInt(filters.price_to)], parse);
        if (filters.game == "")
            client.execute("SELECT * FROM GameDB WHERE " +
                "price>=? AND price<=? ALLOW FILTERING;", [parseInt(filters.price
_from), parseInt(filters.price_to)], parse);
    });
}


var server_callback = function (request, response) {
    console.log("request to: " + request.url + " method: " + request.method)
    if (request.method == "GET") {
        handle_GET(request, response);
    } else {
        handle_POST(request, response);
    }
}
```

db5.html

```
<!DOCTYPE html>
<html>

<head>
    <title>NodeJS+AJAX</title>
    <meta charset="utf-8" />
    <script type="text/javascript">
        var game_filter = "";
        var price_filter_from = 0;
        var price_filter_to = 100;

        function readServer(url, data, callback) {
            var req = new XMLHttpRequest();
            req.onreadystatechange = function () {
                if (req.readyState === 4) {//"loaded"
                    if (req.status === 200) {//"OK"
                        callback(undefined, req.responseText);
                    } else {
                        callback(new Error(req.status));
                    }
                }
            };

            req.open("POST", url, true);
            req.setRequestHeader('Content-Type', 'application/json');
            req.send(data);
        }
        function update() {
            var filters = {};
            filters.game = game_filter;
            filters.price_from = price_filter_from;
            filters.price_to = price_filter_to;
            readServer("/get_table", JSON.stringify(filters), function (err, resp
onse) {
                if (err) document.getElementById("res").innerHTML = err;
                else {
                    var temp = "";
                    temp = "<table cellspacing=\"2\" border=\"1\" cellpadding=\"5
\">\n";
                    var rows = JSON.parse(response).table;
                    for (var i = 0; i < rows.length; i++) {
                        temp += "<tr><td>" + rows[i].game + "</td><td align=\"cen
ter\">" + rows[i].price + "$</td><td align=\"center\">" + rows[i].release_date +
"</td><td align=\"center\">" + rows[i].platform +"</td></tr>\n";
                    }
                    temp += "</table>";
                    document.getElementById("res").innerHTML = temp;
                }
            });
```

```
        }

        window.onload = function () {
            var game_input = document.getElementById("game");
            game_input.oninput = function () {
                game_filter = game_input.value;
                update();
            };
            var fr_input = document.getElementById("from");
            fr_input.oninput = function () {
                price_filter_from = fr_input.value;
                update();
            };
            var to_input = document.getElementById("to");
            to_input.oninput = function () {
                price_filter_to = to_input.value;
                update();
            };
            update();
        }
    </script>
</head>

<body>
    <p>
        HELLO CASSANDRA, I LOVE YOU
    </p>
    <p>
        Game Search<input id="game" type="text"> Game Price from: <input id="from
"
            type="number" value=0> to: <input id="to" type="number" value=100>$.
    </p>
    <p>
        <span id="res" style="font-style: italic"></span>
    </p>
</body>

</html>
```