MATHEMATICS AND NATURAL SCIENCES

Proceedings of the Sixth International Scientific Conference – FMNS2015
10 – 14 June 2015

Faculty of Mathematics and Natural Sciences

VOLUME 1

MATHEMATICS AND INFORMATICS

Mini-Conference "Inquiry-Based Approach in Higher Education in Mathematics and Informatics"

South-West University “Neofit Rilski”
Blagoevgrad

Building Real-Time Web Applications with SignalR and NoSQL Databases

Iliya Nedyalkov
UNWE, Sofia, Bulgaria

Ivo Damyanov
SWU, Blagoevgrad, Bulgaria

***Abstract**: Nowadays real-time web applications are vastly becoming popular. They are widespread in a variety of industries, including: online betting, financial and banking services, online games and many others. The aim of the applications is: to analyze and process big missives of data received by various 3rd parties, store the data which is to be used later on and to only send/push the updates to connected clients (vs. all available clients). In this paper we demonstrate the collaboration between SignalR and NoSQL database for creating a model of a real-time web application matching the needs of the online betting industry.*

***Keywords:** SignalR, real-time web application, NoSQL, big data*

1. INTRODUCTION

Real-time applications are gaining popularity [5]. They have been used in the online betting industry for quite a while- especially within the live betting sections where every situation, point or goal affects the odds of the bookmakers. Microsoft presented SignalR library as a method for creating real-time web applications. These applications have the same goals:

1. Analyze / process the data received;
2. Store the data which is to be used later on;
3. Deliver real-time data / updates to clients.

SignalR allows rapid application development of a real-time web applications which satisfies some of the aforementioned goals including: receiving / analyzing / processing data and sending updates.

Modern web applications still need to store big missives of data received at all times. Real-time systems require even greater efforts and methods for handling data storage and retrieval. The options are:

1. Old-school relational databases;
2. NoSQL databases.

Nowadays all web applications from purely presentational web sites to enterprise information systems are using databases. Ever since the dawn of this technology in the 70s the relational databases have no alternatives and as such, they have been handling the business needs for decades. Internet

had become the utmost used business environment and the web applications must satisfy the following needs:

1. Constant huge increase in the number of users operating
2. Greater missives of data which have to be stored and processed
3. More frequent database queries which have to be executed

However we can now also register the flaws of the relational database, amongst them worth mentioning are: decreased speed for data manipulation upon increasing the volume, low scalability etc.

The web based business is constantly changing and is now searching for alternatives which would match its needs, such as: enhance the speed of the web applications regardless of the increased volume of information and number of users. As a response to these needs the non-relational data bases (NoSQL databases) emerged.

2. EVOLUTION OF DATA DELIVERY TECHNIQUES

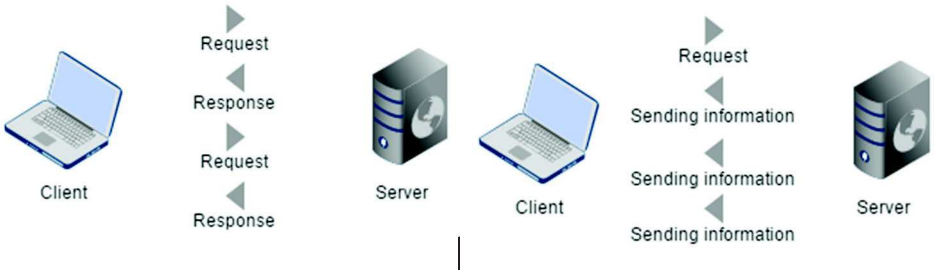
Recently tremendous effort has been invested in developing interactive, real-time multi-user systems. For building a real-time web applications could be applied different approaches: HTTP client pull, HTTP server push, web sockets, etc.

HTTP Client Pull: In the client pull technique, a new HTTP connection is opened every time a document is requested [4]. In a predefined period (Time to Refresh) the client calls the web server to receive updated data regardless of the presence (or the lack of) of any updates. The calls have to be frequent to sustain the information at a real-time pace. Data could be changed ten times for the next 10 seconds followed by a period of no updates in the next 2 minutes - yet again the client will continue to call the server with the same frequency – let's say every second. The main drawback is in the traffic generated by the client's calls and in keeping the server busy to respond regardless of the fact that there might be no updates. Amongst the versions of HTTP client pull could be found:

(i) **HTTP polling** (request → response) – client makes a request and waits for the server to respond [1]. If there is an update it is sent to the client. If there isn't one the server sends an empty response. Figure 1a) depict this communication mechanism.

(ii) **HTTP long polling** (request → wait → response) – Client creates a connection to the server, and it stays opened for a period of time [1]. Within this time slot the client can receive data from the server. The connection is closed right after the server sends its response. Client needs to reconnect after the connection is closed regardless if it's because of a time out or data being sent.

HTTP Server Push: In the server push technique, the connection is sustained until all data is delivered to the client [4]. The push is always initiated by the client. Server sends data to the clients periodically when there are updates. The biggest advantage is that the traffic is reduced tremendously. The communication between the server and the client occurs only when needed (updated information).



a) Request-Response b) Server Push

Fig. 1 Client Pull and Server Push communication models

HTTP Server push mechanism was further improved and standardized and in HTML5 it appear as so called **Server Sent Events** (SSEs). SSEs are sent over traditional HTTP. That means they do not require a special protocol or server implementation to get working. SSEs are handled directly by the browser and the user simply has to listen for messages.

The biggest disadvantage of the push mechanism is that communication is not bidirectional.

Web Sockets: The WebSocket Protocol for a bidirectional communication requires that both the client and the server applications are aware of the protocol details. This means you need a WebSocket-compliant Web page that calls into a WebSocket-compliant endpoint [2].

The client creates a TCP connection to the server which stays opened for as long as it needs. Client initiates the connection. Upon a successful attempt, the server and the client can exchange data in both directions. This is the main difference with Http Server push where the communication after the handshake is only from the server to the client. Using TCP protocol allows avoiding any performance issues.

Microsoft presents SignalR library as a way to simplify the process of adding real-time web functionality to applications. This is the ability to have server code pushing content to connected clients instantly as it becomes available, rather than having the server wait for a client to request new data [3]. SignalR makes possible the development of a brand new type of web

applications that require high frequency updates from the server such as online betting applications.

SignalR uses WebSocket where available, and falls back to older transports where necessary like HTTP long polling.

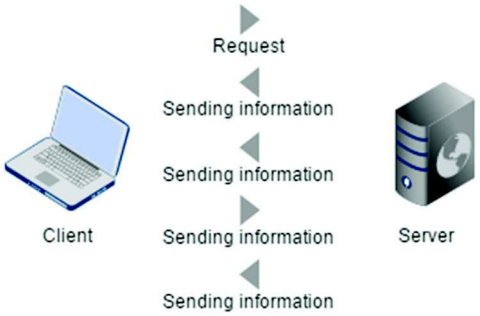


Fig. 2 Bidirectional communication via Web sockets

3. DATABASE HORIZONTAL SCALABILITY

There's a constant increase in terms of demands from the business. They are coming from the necessity to handle more and more users at all times, hence manipulating greater data missives. In order for the applications to meet these demands they must be scalable. The usage of a NoSQL implies **horizontal scalability** which would be applied in the online betting application. Horizontal scalability means adding new servers which are not that powerful and do not use expensive hardware. In general this approach is cheaper compared to a machine upgrade with the latest processors.

Pros: comes at a lower price, easier upgrading. Compared to a vertical scalability at a certain point both the further development and upgrade are no longer possible, i.e. the mother board cannot support further RAM enhancement or there's no better processor (in terms of productivity) available on the market.

Cons: more complicated implementation, necessity of a greater number of licenses, more room needed in the data center.

NoSQL are applicable for horizontal scalability. Approaches/Techniques for horizontal scalability:

Full Replication: In the majority of the cases there are several data base servers which own an identical copy of the data and are being synced. This is also called full replication. A single call is in theory handled by the DB server with the lowest load.

Sharding: Database sharding is a technique for achieving horizontal scalability in the applications. Sharding enables a single logic database to be transferred upon several servers, i.e. one could contain soccer odds, a

second server could contain tennis odds, followed by a different server for basketball odds and so on and so forth. This way the database requests are processed by different DB servers. This distribution of the data among several servers could result in data denormalization. The denormalization however would allow for separate fragments of the database to be independent hence a call to be handled by a single database shard [6].

4. CASE STUDY – ONLINE LIVEBETTING PLATFORM

The online betting industry is amongst the utmost aggressive ones – every point, goal or event during the game could change the bookies' odds. This requires not only a high performance but also a scalable application. To cover these goals we propose creating a SignalR hub which will process upcoming data and stores it in RavenDB shards for later on and also for analyses.

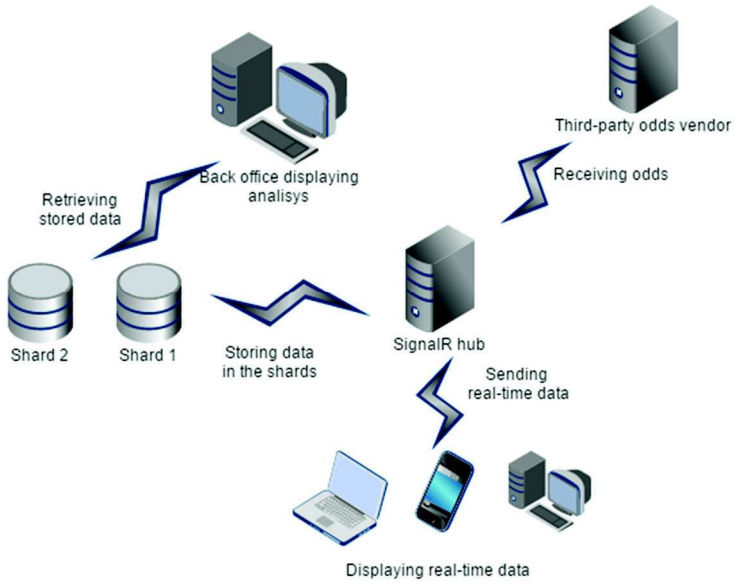


Fig. 3 Live-betting applications with SignalR hub and NoSQL database

Functionality of the application:

1. SignalR hub receives odds from third-party providers in the market.
2. The hub sends only updates or new data to the clients (browsers).
3. The hub also stores the data received in different RavenDB shards. The soccer odds are stored in the first shard, basketball ones in the second one and so on.
4. Pure HTML5 (HTML / CSS / KnockoutJS) page displays the live odds to the clients.

5. MVC back office displays the stored odds and different statistics to operators.

5. CONCLUSIONS

The model of the real-time web application we present in this paper is not the only available solution; however we do believe that it contains the following advantages:

- Using SignalR we gain performance – SignalR is based on TCP transport protocol which allows faster communication compared to other alternatives.
- Using SignalR we reduce the communication between the client and the server. Server sends only updates to the client if any. We skip the requests from the client to the server on a certain amount of time.
- Using SignalR we reduce the load of the server. The server uses the entire CPU resource to analyze and process the data received and sends to the clients only updates.
- Using SignalR we get native client support for desktop and mobile.
- Using NoSQL database we skip any performance problems which we would have had if we were to have a lot of data with the relational databases.

6. REFERENCES

[1] Choudhry, A., Premchand. A. (2014) *Real Time Apps Using SignalR*. International Journal of Computer Trends and Technology (IJCTT) V15(3):92-96

[2] Esposito, D. (2012) Cutting Edge - Understanding the Power of WebSockets, MSDN Magazine (https://msdn.microsoft.com/en-us/magazine/hh975342.aspx)

[3] Fletcher, P. (2014) Introduction to SignalR, (http://www.asp.net/signalr/overview/getting-started)

[4] Gundavaram, Sh. (1996) *CGI Programming on the World Wide Web*, O'Reilly Media

[5] Nedyalkov, I. (2014) *Model of a Real-time application based on SignalR working with big missives of data which is applicable in the gaming industry*, Proceedings of ICAICTSEE – 2014 (in print)

[6] Nedyalkov, I. (2014) *Using Document Databases in building modern web applications based on ASP.NET MVC*, Proceedings of International Scientific Conference – IT in the Business and Education, Science and Economy (in Bulgarian)