Advances in Engineering Software 000 (2017) 1–11

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Robust design to optimize client–server bi-directional communication for structural analysis web applications or services

J. Calvo[a,*], J. Gracia[b], E. Bayo[a]

[a] Department of Building Construction, Services and Structures, University of Navarra, Pamplona, Navarra 31009, Spain
[b] Department of Construction and Manufacturing Engineering, University of Oviedo, Oviedo, Asturias 33003, Spain

ARTICLE INFO

Article history:
Received 20 January 2017
Revised 17 April 2017
Accepted 30 April 2017
Available online xxx

Keywords:
Cloud computing
Web application
Bi-directional communication
Engineering structures
Factorial design
Robust design

ABSTRACT

Current trends in web application development favours bi-directional communication between front-end and back-end applications instead of the traditional ones where the front-end is constantly monitoring the back-end. This way of communication improves the user experience and this work tries to find the best bi-directional way of communication particularized to structural analysis software as a service or web applications. The effects of the most significant factors have been studied to optimize the total time involved in the communication, which is comprised of: time spent sending data from the client to the server, server data processing, and time consumed returning the data back.

Design of experiments (DoE) techniques have been used to characterize the influence of four factors: serialization language, communication protocol, amount of data (size of the structure measured by the number of elements), and server post-processing. Moreover, factors like server workload and network congestion have also been addressed. The first factor is dealt with as a nuisance factor whose influence is to be minimized, and the second one as an uncontrollable variable.

1. Introduction

The software market trend is evolving towards software as a service for consuming products (SaaS) over the Internet. The number of companies following this trend is constantly increasing. Office 365, AutoCad 360 or file storing services like Dropbox, OneDrive or Google Drive, have become successful examples. The SaaS distribution system has many benefits over traditional ones, as explained by Calvo et al. [1], by Subashini and Kavitha [2], and by Miller [3]: high availability through any device connected to Internet, real time software updates and pay-per-use business model. Previous works done by Gracia and Bayo [4], Nuggehally et al. [5], Peng and Law [6], Chen et al. [7,8], Yang et al. [9], and Mackie [10], show that structural analysis software can be provided as a SaaS.

SaaS is similar to traditional client–server applications where each one is responsible for some particular task, therefore optimal communication is crucial for the service to succeed. Moreover, client requirements are beyond one-way communications systems. Current services require that the client will receive data and notifications not only when it is requested but also when the server produces them. Facebook and Twitter are examples of services that make use of bidirectional communication to keep the client up-to-date. They use traditional techniques like Long-Polling or new ones like WebSocket, that is, socket over HTTP protocol.

Nowadays, it is relatively fast for a computer to solve a structural system with thousands of degrees of freedom. This has benefited structural engineers who can develop more complex structural models. However, if we want to move structural analysis software to the internet environment we will have to deal with large amounts of data that need to be sent across Internet. The time data takes to travel from the client to the server, and vice versa, will strongly influence the user experience. Consequently, several decisions have to be made: what processes will be performed in the client and which ones in the server; what type of communications protocol; and what data serialization format, among others. This paper is aimed at providing enough information to choose the best communication technology for structural analysis applications on the Internet environment.

This paper identifies the factors that affect communication time and the technology (or combination of technologies) that is best suited to exchange structural data for Structural Analysis Software as a Service. In this respect, Gracia and Bayo [11] performed previous research focused on conventional client–server communication technology.

In order to simulate user experience, the communication time is measured from client data serialization until data is returned

* Corresponding author.
E-mail addresses: jcalvov@alumni.unav.es (J. Calvo), graciajavier@uniovi.es (J. Gracia), ebayo@unav.es (E. Bayo).

http://dx.doi.org/10.1016/j.advengsoft.2017.04.010
0965-9978/

back from the server. Specifically, the client application stores the data of the structure (nodes, elements, material properties, etc.) as a JavaScript object that needs to be serialized before is sent to the server. Once the data is received, the server relocates the nodes and the elements, and calculates the total weight of the structure (thus considering the server arithmetic operations). Finally, it will serialize the modified structure and send it back to the client, which will parse the data again as a JavaScript object. The client has been implemented following HTML5 standards: JavaScript, CSS and HTML, while the server has been developed with PHP, a well-known server-side programming language designed for web applications.

This DoE is mainly focused on four primary factors: serialization language, communication protocol, size of the structure and server post-processing. Server workload (noise) and network congestion are also considered but as nuisance factors. Factors like the web browser and the web server are not relevant for this study and have been considered as constant factors. To identify parameter interactions a full factorial design is chosen. Finally, robust parameter design (RPD) techniques are applied to minimize the influence of the noise factor. Different models of DoE techniques are summarized by Tanco el al. [12].

In the following sections, the factors analysed in the experimental program are identified. Then, the application developed to measure response times is introduced. Finally, and after following DoE methodology, an analysis of the results is presented.

2. Experimental program

This section describes the experimental program that has been developed to identify the best bi-directional communication setup for web services (or web applications) specialized in the analysis and design of structures. As mentioned before, several tools and protocols are studied by means of a DoE. One of the main difficulties of this experimental program is to deal with uncontrollable factors.

As stated before, in order to develop structural analysis software on Internet several software architectural decisions have to be made. One of these is which processes are implement on the client side (front-end) and which ones on the server side (back-end). If heavy processes are implemented at the server side the computer hosting the server should be able to handle large amount of simultaneous user requests and also to resolve each request as quickly as possible, so that the user experience is not compromised. The main advantage of this architecture is that users do not need high performance computers, since heavy processes are executed at the server. Conversely, if these processes are executed at the client side, the response will be fastest and the server workload will decrease accordingly. User experience will be improved and cheaper servers would be enough to host the web application. However, the main disadvantage is that the code is accessible by users, in some sense it becomes public. One option is to obfuscate the JavaScript code but it will continue to be accessible.

In this research, the first option is adopted and therefore heavy processes are implemented at the server side along with client and server exchanging all the information relative to the structural models. In this way, communication times will be measured to find which technology performs better.

2.1. Objective and variables

Experimental software has been developed to automatize the measurements and the testing needed to identify the influence of the factors mentioned above. This software allows the selection of a combination of levels for each primary factor and to simulate client–server communications. It also simulates the workload at the server side.

The total time consumed in a full communication cycle is selected as the response measurement for each experiment. Time becomes critical when identifying the efficiency of each factor and when assessing the user experience. The time is measured in milliseconds and the main target of the DoE is to identify the factors that minimize it.

The experiment factors can be classified into three types: primary, constant and nuisance. The former ones are factors whose influence is unknown and are selected as the main target of the experimental program. Constant factors can also affect the experiments but they are not selected as targets of the research. Nuisance factors are not interesting for the research; they are uncontrollable and can modify measurements unexpectedly. Nuisance factors that can be controlled in the experimental setup are referred to as noise factors.

Four primary factors have been identified in this experimental program: serialization language, communication protocol between client and server, data size of the engineering structure and server data processing. The main constant factors that will not be studied are: web browser (it has an influence but does not determine the best primary factors for optimal bi-directional communication), server programing language, and web server application. Finally, there are two factors that affect the time measures but are uncontrollable: network congestion during experiments and server workload. The last one will be treated as a noise factor.

Hereafter a thorough explanation of each primary factor that has been studied will be described.

2.1.1. Serialization language

Each application defines its own data model as arrays, objects, databases or any other data structure. However, in order to communicate with other applications, this data model has to be serialized. This process translates data into a format that can be stored or transmitted across the network so that it can be reconstructed in the same or another computer (the server in this case). The main drawback is that data is commonly translated into ASCII strings. The ASCII representation of numeric values is larger than the binary representation. However, serialized data is standard while binary data has to be transformed depending on whether the destination machine is low endian or big endian.

The most widely supported languages in SaaS are XML (eXtensible Markup Language) and JSON (JavaScript Object Notation). Wang [13] provided an interesting comparison between these two languages. In this research, a less popular serialization language named YAML (recursive acronym for “YAML Ain’t Another Markup Language”) has been also studied. Fig. 1 shows a small example of the syntax of these three languages.

XML is the most widely supported. It was design as a simple, very flexible, text formatting language. It is playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. There are a variety of APIs to access XML data; some of them have been standardized. Nonetheless, XML support has been implemented in different ways on each browser and as a consequence performance is not constant among them. It is extremely compelling, compared to JSON and YAML, when dealing with complex data structures. However, its heavy use of mark-up tags tends to transform data structures into large strings, increasing the amount of information to be transferred.

Meanwhile, JSON is gaining adepts in web development environments, and it has a huge support in most browsers. Its syntax is similar to JavaScript, key/value pair notation is the mechanism to store and read data. Data serialised in JSON is lighter than in XML: it represents the same amount of data with less characters. Front-end web developers used it to speed up the development of

Please cite this article as: J. Calvo et al., Robust design to optimize client–server bi-directional communication for structural analysis web applications or services, Advances in Engineering Software (2017), http://dx.doi.org/10.1016/j.advengsoft.2017.04.010