Software Architecture for Efficiently Designing Cloud Applications using Node.js

Ho-Kyeong Ra[1], Hee Jung Yoon[1], Asif Salekin[2], Jin-Hee Lee[3],
John A. Stankovic[2], Sang Hyuk Son[1]
[1]Department of Information and Communications, DGIST
[2]Department of Computer Science, University of Virginia
[3]CPS Global Center, DGIST

ABSTRACT

Node.js is a prevalent software platform used for scalable server-side networking applications on the cloud. To fully exploit the potential of Node.js when developing cloud applications, the implementation process should be eased and supported. There are numerous existing guidelines that enable the use of Node.js for application development, however, many are extensive and complex, even for general purpose prototyping. We propose a practical solution for cloud application development using Node.js and Express library by presenting: (1) a software architecture which utilizes two standard inheritance pattern techniques, the top-down and divide and conquer approaches, to effectively organize the structure of the application for improved maintainability and extensibility in the long-run, and (2) an easy-to-follow guideline that instructs the implementation procedures for developing Node.js cloud applications. In this paper, we briefly describe our architectural design and examples of cloud application development in different fields of study.

Keywords

Software architecture; Cloud applications; Node.js

1. INTRODUCTION

The increasing interest is various Internet of Things (IoT) applications, have led to the generation of various data points collected from a number of heterogeneous devices to be gathered in a single repository for application development. The distributed nature of these systems allowed researchers to shift to cloud infrastructure for achieving novel application designs with the support of integrated data processing and effective resource management.

Despite its attractiveness however, implementing such cloud-based applications is not in any way trivial. Designing an effective cloud application askes the developer to have knowledge of various technical domains from server operation, local and web-application languages, to data communication protocols. For example, for an effective web development, developers typically require knowledge of up to five different programming languages, such as JavaScript, HTML, CSS, a server side language such as PHP, and SQL [1].

In overcoming these challenges, a recently introduced software platform called Node.js [2] has gained a significant amount of traction in the developer communities since its introduction in 2009. Specifically, Node.js is a scalable single-threaded server-side JavaScript environment implemented in C and C++ [3]. Despite the recent development, Node.js's approach in developing web applications have made it an attractive alternative to more traditional platforms such as Apache+PHP and Nginx servers.

One of the many benefits of Node.js is its architecture that makes it easy to use as an expressive, functional language for server-side programming [5]. Although it may be trivial to perform development in Node.js once a developer fully comprehends the language, for beginners to use Node.js effectively, there are many obstacles prior to being able to implement real cloud-based applications. A study states that the platform as a whole lacks code quality standards and reasoning through a file which has been edited by various developers can be challenging [4]. Moreover, due to the framework being relatively young and the software yet reaching maturity, compared to traditional frameworks, developers face the lack of documentation support and can face troubles in receiving support from the development community.

To the best of our knowledge, there has not yet been a Node.js software architecture or implementation guideline, where developers are influenced to create well-structured applications. Nevertheless, as software developers, we should be aware of the fact that organizing the program design with solid boundaries is crucial in a successful real-world deployment. Designing an ad-hoc model will cause logical complexity and difficulties in maintaining and updating the application over time. Moreover, without good guidelines, it takes a significant amount of time and needless effort to even configure the basic application development environment.

The contributions of this work are three-fold:

- A software architecture that tackles the challenges

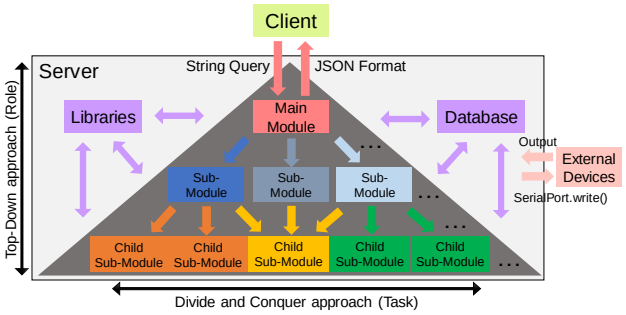

Figure 1: Overview of our software architecture.

of maintaining and updating cloud applications developed using Node.js. This design layout supports the resulting application to comprise well defined, independent components which lead to better maintainability. In addition, new capabilities can be added to the application without major changes to the underlying architecture.

- Implementation guideline that provides explicit, but straightforward instructions for developing general purpose Node.js prototype. This guideline instructs how to: (1) create prototypes by using a limited set of APIs, (2) divide modules and organize functions, (3) allow a client to communicate to a server, (4) utilize cloud application database, and (5) handle uploading and downloading files. Through this guideline, developers can focus on implementing application logic.

- Application examples that open the possibilities for new cloud applications to be developed. Our software architecture and guideline can be the basis of developing a variety of cloud applications. We provide examples of application scenarios that can be enabled by our work.

2. ARCHITECTURE

We detail two software development processes that make up our software architecture. The top-down approach helps developers to couple and decouple modules. This supports developers to observe application flow from the main module, which plays a role of a main class in Java, to other sub-modules in a top-down manner. It is essentially the breaking down of the application development to gain insight into its compositional sub-modules. On the other hand, the divide and conquer approach guides developers to divide tasks into simpler modules while enabling multiple module development concurrently. We use the top-down approach first and then the divide and conquer approach in the later step so that the roles between the modules are first defined before tasks are assigned to each roles as shown in Figure 1.

3. APPLICATIONS

There are various types of cloud applications that can take advantage of our proposed software architecture and implementation guideline. Examples include applications that simply log information to cloud through web, exchange medical or healthcare information using mobile application, and execute physical component actuations at local ad-hoc network. In general, our work can be applied to cloud application developments that require the basis of setting up fundamental requirements. These requirements include sending and receiving data, having a database to store information, calling external executables for machine learning algorithms, and ensuring space for the application to be expanded.

4. CONCLUSION

We propose a software architecture and implementation guideline for developing Node.js cloud applications. Our architecture uses the top-down and divide and conquer approaches to effectively structure application design for better maintainability and extensibility over time. On the other hand, our guideline instruct implementation procedures for straightforward development of these applications. We also present practical scenarios of how developers can utilized our software architecture and implementation guideline for different types of applications. As part of our on-going work, we apply this architecture on an medical application that provides an advice and alarm infrastructure based on collected data and parameters set by healthcare providers. For future work, we plan to further extend our studies by applying this onto different types of applications to show the effectiveness of our architecture. We envision that this work would influence both developers and non-technical researchers to efficiently and easily develop cloud applications.

Acknowledgments

This research was supported in part by the DGIST Research and Development Program of the Ministry of Science, ICT and Future Planning of Korea (CPS Global Center), and the ICT R&D program of MSIP/IITP (14-824-09-013, Resilient Cyber-Physical Systems Research).

References

[1] S. Frees. A place for node.js in the computer science curriculum. *J. Comput. Sci. Coll.*, 30(3):84–91, Jan. 2015.
[2] Joyent, Inc. Node.js. `http://nodejs.org/`, 2016.
[3] R. R. McCune. Node. js paradigms and benchmarks. *Striegel, Grad. Os. F.*, 2011.
[4] Souci, Benjamin San and Lemaire, Maude. An inside look at the architecture of nodejs. `http://mcgill-csus.github.io/student_projects/Submission2.pdf/`, 2014.
[5] P. Teixeira. *Professional Node. js: Building Javascript based scalable software.* John Wiley & Sons, 2012.