A Study Case of Restful Frameworks in Raspberry Pi: A Perfomance and Energy Overview

Luiz H. Nunes *, Luis H. V. Nakamura *, Heitor de F. Vieira *, Rafael M. de O. Libardi *,
Edvard M. de Oliveira *, Lucas J. Adami *, Julio C. Estrella *, Stephan Reiff-Marganiec †
* *University of São Paulo (USP)*
Institute of Mathematics and Computer Science (ICMC), São Carlos-SP, Brazil
Email: {lhnunes, nakamura, heitorfv, mira, edvard, ljadami, jcezar}@icmc.usp.br
† *University of Leicester*
University Road, Leicester, LE1 7RH - UK
Email: srm13@le.ac.uk

***Abstract*—This paper analyzes the execution behavior of web services on devices with limited resources. The experiments compare web services in the Axis2 and CXF frameworks analyzing performance and power consumption. To determine which framework is better suited for service provision, a testing environment and a performance and energy evaluation between them are presented. We show that the Raspberry Pi can be useful in service-oriented applications for different types of tasks. Bringing together the best features of small devices and SoC, it is possible to provide diverse, mobile and green applications.**

***Keywords*-Web Services, Service-oriented Architecture, Quality of Service, Performance Evaluation, Apache Axis2, Apache CFX, Raspberry Pi**

I. Introduction

The combination of embedded devices, sensors and the Internet allows to link the physical world with the cyber space, expanding the Internet to the Internet of Things (IoT) [3]. The increasing use of mobile and embedded devices jointly with the expansion of IoT requires studies about which tools, protocols, frameworks, and applications should be used in order to increase efficiency of the systems while reducing costs. Additionally, the economy of resources is a recurrent goal in much recent work.

Due to service-oriented applications using other resources to perform tasks, they can overcome these limitations and improve the performance of embedded devices, including the Raspberry Pi. However, the use of service-oriented applications in embedded devices still need to be investigated with respect to other types of information, such as larger messages that demand more resources. This study is very relevant because services use large amounts of data to create additional information in the message body, which increases the size and processing time of those messages [4]. Another important factor to investigate is the energy consumption to perform such processing in embedded devices. In this paper, we present a performance and energy consumption study evaluating the behavior of RESTful web services using the Raspberry Pi. To perform this evaluation, services with the same features were developed using the Axis2 and CXF frameworks developed by the Apache Software Foundation.

This paper is organized as follows: Section II presents a literature review of embedded web services performance evaluation studies and methodologies. Section III highlights the characteristics of frameworks and technologies used in this study. Section IV describes the methodology and configurations used for the experiments. The results are then discussed in Section V. Finally, the conclusions and directions for future work are presented in Section VI.

II. Work Related

Previous web services performance evaluation methodologies for embedded and mobile environments were categorized into two approaches: real device experiments and simulation based on mathematical models. Real experiments were found in [8], [9], [6] and [4] and are the most used. In these approaches, real devices and prototypes are used to measure response variables in a real environment using replications for non-deterministic variables.

Although this is the widely used approach, it is quite expensive, needs to follow a rigid methodology and requires deep statistical analysis. Another methodology found in [10] and [5] uses mathematical models and simulation tools to evaluate proposed architectures. In these approaches, mathematical models representing the system to be evaluated are created. However, there is a risk that the model is not realistic and that the environment is not truthfully captured. We used real device experiments using a testbed environment for this study.

As we know from the literature, RESTful are better than SOAP for mobile devices, because they save a lot of resources and energy and generate less overhead in data transfers and spend less time to pack/unpack the request and response messages. Nevertheless, there are different ways to implement and deploy a RESTful web service. Thus, this paper aims to complement these works to present an

energy consumption and performance overview of RESTful frameworks in Raspberry PI.

III. Frameworks

A. Axis2 Framework

Apache Axis2 is a project implemented in the Java language that facilitates the implementation of web services for both client applications and service providers. Besides, it offers a completely object-oriented approach and it is built upon a modular architecture. The messages used in this framework are built from the Apache Axiom library, which provides a set of XML methods for the creation and organization of objects into a tree [2]. Sending and receiving messages is one of the key tasks of web services. The Axis2 architecture provides two pipes (or flows) to perform these two operations. Therefore, the complex message exchange (MEPs) are performed through the combination of these two flows [7].

B. CXF Framework

Apache CXF is an open source framework that provides easy development of web services in Java, for both SOAP and RESTful architectural concepts. This framework uses the Java API for RESTful Services (JAX-RS) for RESTful architectural concepts, which provides the semantics for the creation of RESTful web services and abstracts implementation details of their clients. The message flows in CXF are handled by interceptors that perform a particular functionality. They can be added to an interceptor chain, which in turn are grouped and arranged in stages. The interceptor chain manages resources and information from others CXF components (Front-End, Protocol Bindings, Transports) to handle the message and establish communication among client and service [1].

IV. Performance Evaluation

Figure 1 illustrates the messages flows between the applications used in this paper (steps are marked 0 to 6).

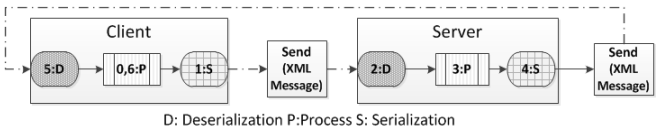

Figure 1: Messages flow in the experiments.

The client device starts the application by generating a sequence of random numbers and then performs the serialization of this sequence into an XML message. Once built, this message can be sent to a web service provided by the server device. When the message arrives, the server first deserializes the message, and then process its content sorting the numbers that were received from the client, so a new message with the ordered numbers is created, serialized in XML format and sent back to the client device.

Finally, the client deserializes the response from the server and ends its execution. To monitor the energy consumption, we used Arduinos Nanos because they are capable to ensure accuracy and synchronism in data collection through an automated code to start and finish the measure. The Arduino was used to collect data like time, voltage and current. These data are collected 170 times per second from both client and server devices through a shield that was made to measure the energy, and are stored in a desktop computer using the Mini-B USB interface.

A. Experiment Enviroment

Two Raspberry PI devices model B interconnected by a gigabit switch. The configuration uses the standard CPU clock (700Mhz) and a class 4 SD card with 8Gb. The operational system used was the Raspbian GNU/Linux with JDK (Java Development Kit) 1.6. The memory in JVM (Java Virtual Machine) was set to 128Mb.

B. Experiment Design

The experiments were designed to gather as much information as possible to compare the performance and energy differences between two different development *Frameworks* for web services (Axis2 and CXF) with light (100Kb) and heavy (500Kb) messages. Four experiments involving combinations of factors (Framework and Message Size) with different configurations or levels were designed. Each experiment was repeated 50 times in order to calculate the average time and guarantee a statistically correct result. Besides, the standard deviation and confidence intervals (assuming a 95% confidence level) were also calculated for each of the average times collected.

V. Results

Table I shows the total time (T.T) and confidence interval (C.I) in seconds to perform the request in server and client side respectively. In both sides, the total time was proportional to the size of the message, as the processing time is directly proportional to the amount of data. Server side shows that CXF framework requires less time than the Axis2 framework to execute services with 500Kb messages. This reduction is explained by the use of the JAX-RS library, which handles RESTful messages directly, whereas Axis2 has an overhead to convert REST to SOAP messages during service execution.

Table I: Total Time of Server and Client Devices

Total Time								
Side	**Server 700MHz**				**Client 700MHz**			
M. Size	**100Kb**		**500Kb**		**100Kb**		**500Kb**	
	T.T	**C.I**	**T.T**	**C.I**	**T.T**	**C.I**	**T.T**	**C.I**
Axis2	15.48	0.29	84.79	0.47	30	0.33	128,15	0,53
CXF	18.39	0.13	81.62	0.59	38.64	0.14	125,22	0,68

Client side considers serialization and deserialization times, the network traffic, and the service execution time

on the server side. We can verify that Axis2 framework had a better performance only for 100Kb messages while CXF framework had a better performance for 500Kb messages. JAX-RS libraries use methods of generic classes to handle RESTful messages which has better performance for larger messages. On the other hand, Axis2 framework handles REST messages as SOAP messages in its core which causes an overhead proportional to the quantity of converted data.

Table II shows the total energy consumption (T.E.C) and the C.I during the service execution in server and client side respectively. Results shows that CXF consumes equal or less energy than Axis2, except in 100Kb messages. Thus, in general CXF framework presents better performance and energy results than Axis2 for those experiments.

Table II: Energy Consumption of Server and Client Devices

Energy Consumption								
Side	**Server 700MHz**				**Client 700MHz**			
M. Size	**100Kb**		**500Kb**		**100Kb**		**500Kb**	
	T.E.C	**C.I**	**T.E.C**	**C.I**	**T.E.C**	**C.I**	**T.E.C**	**C.I**
Axis2	31.98	0.57	164.41	1.27	58.73	0.62	233.87	1.31
CXF	35.01	0.32	157.68	1.62	70.72	0.25	223.66	1.1

VI. Conclusion

The Raspberry Pi offers a computational architecture with a general purpose, low cost, and low power consumption. In this paper, the performance and energy results showed that Raspberry Pi resources are insufficient for the execution of tasks that depend mostly on the processor. The CXF framework proved to be more suiteble in this environment, as it provides support for the creation of both service providers and client applications. Furthermore, when using CXF in service provider applications it achieves shorter overall times for large messages (500Kb) while Axis2 is better for small messages (100Kb).

In future work we intend to monitor the energy consumption using batteries and also testing another kinds of applications, like IO-Bound services to verify the performance of reading and writing data into SD card. We also intend to monitor the energy consumption using batteries and evaluate other kinds of applications (IOBound and MemoryBound), devices (Parallela, BeagleBone, etc.), frameworks (Jersey, Restlet, etc.) and protocols (Constrained Application Protocol (CoAP)).

VII. Acknowledgements

This project was financially supported by São Paulo Research Foundation - FAPESP (processes 11/09524-7, 2013/26420-6 and 2011/12670-5) and National Council for Scientific and Technological Development - CNPq (process 133841/2012-0).

References

[1] Apache cxf - how it works. Avaliable in http://cxf.apache.org/docs/custom-transport.html. Last access: 11/04/2014.

[2] Welcome to apache axiom. Avaliable in http://ws.apache.org/axiom/index.html. Last access: 11/04/2014.

[3] P. fei Fan and G. zhao Zhou. Analysis of the business model innovation of the technology of internet of things in postal logistics. In *Industrial Engineering and Engineering Management (IE EM), 2011 IEEE 18Th International Conference on*, volume Part 1, pages 532–536, 2011.

[4] C. Groba and S. Clarke. Web services on embedded systems - a performance study. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pages 726–731, 2010.

[5] H. Hamad, M. Saad, and R. Abed. Performance evaluation of restful web services for mobile devices. *Int. Arab J. e-Technol.*, 1(3):72–78, 2010.

[6] M. Jansen. Evaluation of an architecture for providing mobile web services. *International Journal On Advances in Internet Technology*, 6(1 and 2):32–41, 2013.

[7] D. Jayasinghe. *Apache Axis2 Web Services, 2nd Edition*. Packt Publishing, February 2011.

[8] R. Mizouni, M. Serhani, R. Dssouli, A. Benharref, and I. Taleb. Performance evaluation of mobile web services. In *Web Services (ECOWS), 2011 Ninth IEEE European Conference on*, pages 184–191, 2011.

[9] A. Papageorgiou, J. Blendin, A. Miede, J. Eckert, and R. Steinmetz. Study and comparison of adaptation mechanisms for performance enhancements of mobile web service consumption. In *Services (SERVICES-1), 2010 6th World Congress on*, pages 667–670, 2010.

[10] Q.-D. Vu, B.-B. Pham, D.-H. Vo, and V.-H. Nguyen. Towards scalable agent-based web service systems: performance evaluation. In *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, iiWAS '11, pages 481–484, New York, NY, USA, 2011. ACM.