Evaluation of an XML Database Based Resource Directory Performance

Stevan Jokić, Srdjan Krčo, Jelena Vučković, Nenad Gligorić, Dejan Drajić

Abstract — **In this paper, an implementation of a Resource Directory, a directory storing descriptions of resources available in an Internet of Things system, based on a native XML database is described. Its performance is compared with performance of a Resource Directory with the same functionality, but implemented using a relational, SQL database. The evaluation results show hat the XML based Resource Directory provides more flexible management of the resources and shorter lookup execution time, while SQL based Resource Directory provides shorter response times.**

Key words — **Native XML databases, relational databases, IoT, Future Internet, Resources.**

I. Introduction

THE technological progress is all about the low cost microprocessors and embedded devices that enables the new ways of communication, making the concept of large set of devices connected to Internet feasible. This concept is going to change the perspective of future application in respect to large number of information available from tiny low powered devices. In general, all of these scenarios are possible due the communication between the machines (M2M), actuators, sensors, and many other "things". The concept is globally gathered under the one name - Internet of Things (IoT).

In order to impose the new addressing schemes, many of the existing standards are replaced with the light-weight versions. Therefore, evaluation of databases, transport protocols, and message formats are required to leverage the better solution for different IoT scenarios. In this paper is considered M2M resources storing. Next chapters will show the existing relational database resource storing implementation and comparing to the native Extensible Markup Language (XML) database storing. At the end of the paper are presented comparing results and conclusion about flexibility and performances of the relational and XML based resource storing.

This paper describes work undertaken in the context of the SmartSantander project. The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement n° ICT-2009-257992.

Stevan.J, Ericsson Serbia, University of Novi Sad, (e mail: stevan.jokic@ericsson.com).

Srđan. K.. Ericsson Serbia, University of Belgrade, e-mail: srdjan.krco@ericsson.com

Jelena V., Ericsson Serbia, e-mail: jelena.vuckovic@ericsson.com

Nenad G., Ericsson Serbia, University of Belgrade, e-mail: nenad.gligoric@ericsson.com

Drajić D., Ericsson Serbia, e-mail: dejan.drajic @ericsson.com

II. Resource Directory

Resource Directory (RD) is originally designed and implemented during SENSEI project [1]. The main objectives of the SENSEI project were integration of the Physical with the Digital World of the Network of the Future.

All entities in such an integrated system were considered as resources. Resource Descriptions are introduced in the SENSEI project as human/machine understandable representation of the Resources and are formatted following an agreed XML schema. Each Resource Description is represented as an XML structure that contains a set of tags describing the Resource, as well as the URL where the so called Resource Endpoint (REP) - access point to the Resource is located. A REP provides the Resource Access Interface (RAI) for accessing the Resource. Following XML is a resource description for a gas sensor on deployed on a bus in Pančevo as a part of the ekoBus system [ref].

```
<Resource-Description>
  <Resource-ID>
urn:sensei:ericsson.com:EnvironmentalSensors:bu
sgps:358278006369805
  </Resource-ID>
  <Name>Bus Location Sensor</Name>
  <Tag>Bus</Tag>
  <Tag>GPS</Tag>
  <Tag>Sensor</Tag>
  <Tag>Tracker</Tag>
  <Tag>Pančevo</Tag>
  <Tag>24</Tag>
  <Tag>358278006369805</Tag>
  <RAI-Description>
    <Description>GET returns sensor values
    (RDF)</Description>
    <REP-Locator>
      http://www.ekobus.rs/rephandler/gps/35827
      8006369805
    </REP-Locator>
  </RAI-Description>
</Resource-Description>
```

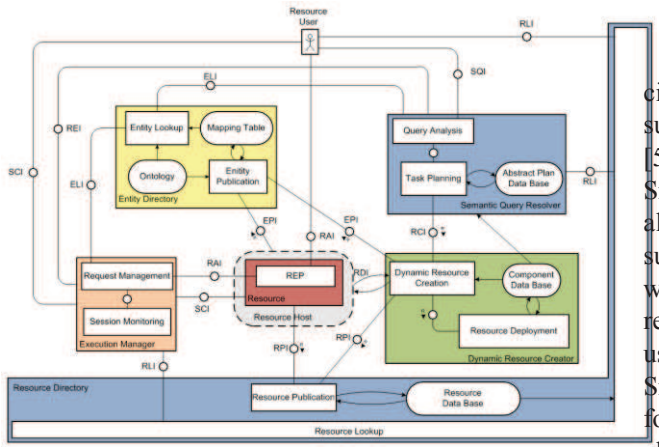

Fig. 1. RD in SENSEI architecture

RD contains resource descriptions of all available resources. Management of descriptions stored in a RD is possible via a set of RD's interfaces which provide Resource Create Read Update Delete (CRUD) as well as Resource querying and subscribing capabilities.

The role of the RD is to make the glue between Resources which advertise the operations they offer as Resource Descriptions and potential clients that look for particular functionalities. Its place in the overall SENSEI system is presented in Fig. 1.

The main components comprising an RD are the following:

• Resource Publication Interface (RPI) - responsible for publication of available resources and their descriptions.

• Resource Lookup Interface (RLI) - based on the received search parameters identifies suitable resources in the Resource Database. Subscribing interface is implemented as part of the RLI interface.

• Resource database - stores descriptions of all resources.

The RD functionality was implemented in JAVA programming language in combination with a MySQL database for persistent storage of data. Interaction with the resources and the users was based on the RESTlet framework [3]. The Restlet approach utilizes the basic HTTP methods (POST, GET, PUT and DELETE) to build CRUD (Create, Read, Update and Delete) applications. All requests to RD and RD responses are XML formatted messages.

Evaluation of the main technical features of the SENSEI architecture was performed in the ecoBus system [4]. The ecoBus system utilizes public transportation vehicles to carry a set of sensors across the city of Belgrade to observe a number of environmental parameters as well as events and activities in the physical world. Resources in this system are: GPS, environmental sensors on buses, bus lines, line path and estimated bus arrival time service. All resources are stored in RD using appropriate resource descriptions. Clint access to the system is enabled by web and android application which uses RD's RLI interface to find appropriate resources, and their REP for data and measurements accessing.

III. RD ROLE IN THE SMART SANTANDER PROJECT

Smart Santander project proposes a unique in the world city-scale experimental research facility platform in support of typical applications and services for a smart city [5]. Management of a highly versatile set of SmartSantander resources is entrusted to the RD. The RD also provides notification managements for resource subscribers. The subscribers are platform end users, as well as other SmartSantander components. Many of the resources in SmartSantander could not be described well using the SENSEI resource description schema. Wrapping SmartSantander resources to the SENSEI defined schema for resource descriptions results in complicated *Tag* elements. The following XML code is a part of the resource description of the Wisebed CTI-Testbed:

```
<Tag>node_name #
urn:wisebed:ctitestbed:0x1bee</Tag>
<Tag>factory_class #
de.uniluebeck.itm.tr.runtime.wsnapp.WSNDeviceAp
pFactory</Tag>
<Tag>node_type # telosb</Tag>
<Tag>node_port # XBPTDXR1</Tag>
```

From this example it's noticeable that the "key-value" data are stored in the text content of the Tag element, separated by hash symbol and hence cannot be used without prior processing. Resource wrapping degrades presentation of XML data which is one of the key XML advantages. Further to that, some parts of the wrapped resource descriptions must use the CDATA XML's sections in order to retrieve XML validity [6]. Resource wrapping also degrades RD's Tag querying capabilities, which is used for resource lookup as well as for resource subscription.

RD's subscribing capabilities are used extensively in the SmartSantandar project. This procedure involves describing resources using wrapped tags and submitting them to the RD's subscribing interface. The subscribers receive notifications from the RD about the changes regarding the subscribed set of resources. These notifications also contain wrapped resource descriptions. This requires implementation of the custom, hybrid XML and text parsers on the subscribers' side thus adding complexity.

To address the issues above and simplify the problem of storing versatile XML files, we implemented a version of RD using a native XML database (SEDNA [ref]) as permanent storage.

They use different data types, and XML's ability to change structure in the middle of a document does not mesh with a relational database's rigid table structures. XML generating, in dependency to the data modeling complexity could retrieve data from many tables. Also, when an XML document is stored in a relational table, information can be lost, such as element ordering and the distinction between attributes and elements.

IV. SEDNA XML RD IMPLEMENTATION

Sedna is an open-source native XML database system being developed by the MODIS team at the Institute for

System Programming of the Russian Academy of Sciences [7]. Sedna is developed from scratch in C/C++ and Scheme, also implements W3C XQuery language and its data model exploiting techniques developed specially for this language. XQuery is a query and functional programming language that is designed to query collections of XML data [8].

The Sedna based RD is implemented in Java programming language as a web application. RESTLET libraries [10] are used for the REST web services implementation. Connection to the Sedna database is maintained using Sedna Java driver API library. Data are separated to the appropriate database collections. Collections can be considered as tables in the relational database terms. The collections stores and manages XML files. The resource descriptions are stored in the collection named "*rdcoll*". Subscriptions are stored in the "*subscriptions*" collection, resource matching the subscriptions in the "*matched*" collection and notifications for subscribers are stored in the "*notifications*" collection.

The XQuery is used in the querying procedure. This enables more powerful lookup capabilities compared to the tag based lookup used in the original RD implementation. The old tag based lookup capability is kept as well as it's extended to support XQuery client lookups as well as mixing the tag and XQuery lookups.

Resource publishing, reading, updating, deleting is performed using Sedna Java driver API applied to the appropriate collection in database. In the Sedna based RD implementation resource descriptions are addressed using provided Resource-ID element. If the Resource-ID is omitted during publishing procedure, a Sedna generated ID is added to the resource description. A resource reading from the native XML database does not require XML serialization from several tables, because RD's database already has appropriate formatted resource description. All this simplifies resource managing procedures because there is no need to manage several tables during resource description storing, reading, updating, deleting. Access to resources is provided through the appropriate URL in the following format RD_URL/rd/{Resource-ID}.

The subscription interface is separated from the RLI. A subscriber can use both XQuery as well as tag based queries. The original subscription capability is extended to enable subscribers to read notifications using the HTTP's GET method. This approach is useful in the situations when subscribers couldn't provide inbound HTTP connections. Notifications can be automatically deleted after reading, if appropriate URL parameter is used at the access time.

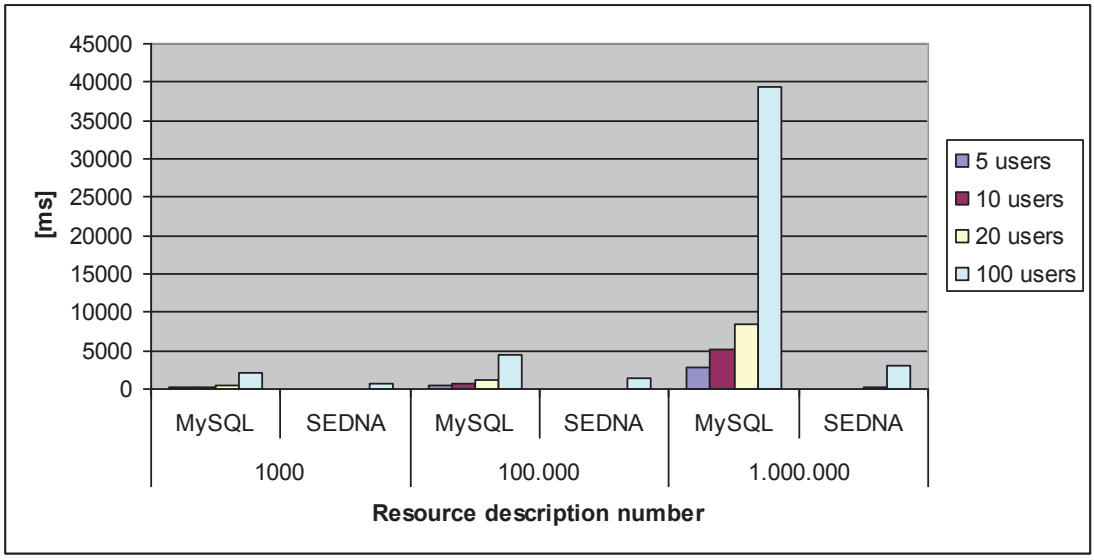

Fig. 2. Response time comparing for various number of simultaneous requests and 1000, 100.000, 1.000.000 resource descriptions in the RD respectively

V. MySQL and SEDNA RD Performance evaluation

Evaluation of the response times for the main RD features is presented in this section. The response times are collected using the Apache JMeter tool [9]. Apache JMeter is an open source Java desktop application designed to load test functional behavior and measure performance. Of a system The JMeter provides a rich set of the components available using Graphical User Interface (GUI). This set contains HTTP GET, POST, PUT, DELETE methods as well as a thread manager for multi user testing, a number of helping components like timers, counters, asserts, etc. Test components could be executed simultaneously using a number of threads. Complex test scenarios can be created in the JMeter tool using XML formatted test plan files.

The firs set of tests is performed for a variable number of resource descriptions in the databases and a variable number of simultaneous requests. The tests are performed for 1000, 100.000 and 1.000.000 resource descriptions and 5, 10, 100 simultaneous requests. Resource descriptions are the same, with the exception of the Resource-ID. The test results are shown on the Fig. 2. Performed operation

during the RD testing was default resource description listing. This operation returns a set of resource descriptions limited to length of 10. It can be noticed that the Sedna based RD implementation has significantly shorter response times in all test conditions. In the Sedna based RD implementation resource listing and response limiting is implemented using Sedna API for collection management. The main reasons for the longer response times of the relational based RD implementation is primarily due to the need to query several tables and to serialize the relational database response to XML format.

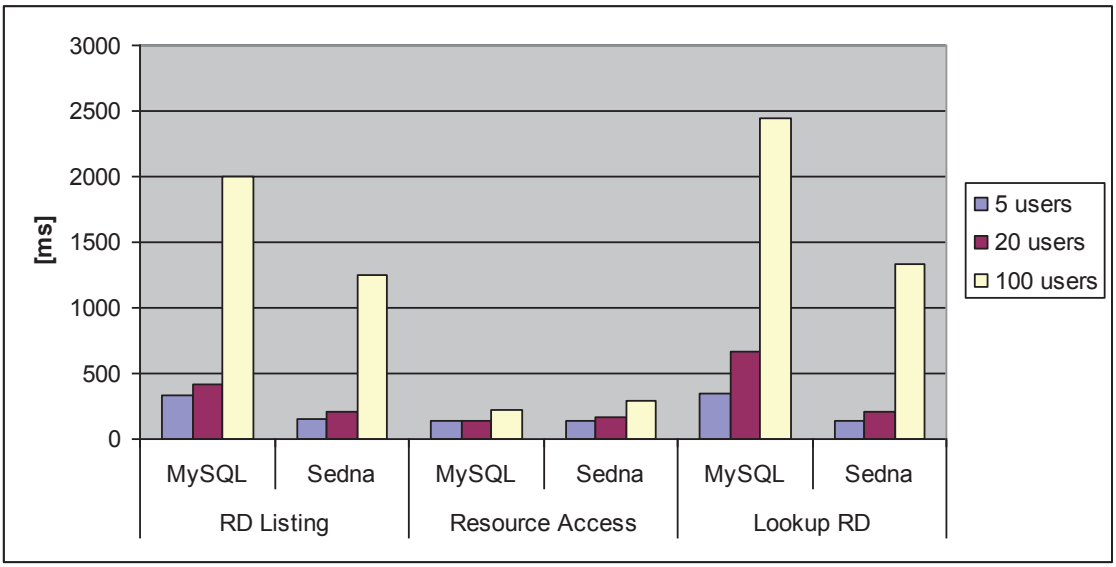

Fig. 3. Response time comparing for EcoBus RD data

The second set of tests is performed on the real RD data from the EcoBus system. The most frequently operations used in the system are evaluated. Resource descriptions from the EcoBus system are copied to the XML based RD and both RDs are hosted on the same server. The test results are shown in Fig. 3. Tests are performed for the operations set the most frequently used in the EcoBus system: resource listing, lookup and access to the particular resource description. The lookup was performed for the tag *Bus,* with 70 matched resource descriptions in the RDs. It can be noticed that the XML based RD has shorter response times except in the case of accessing a particular resource description. In that case the relational based RD has slightly shorter execution time. The most frequently executed RD operation in the EcoBus system is lookup, regarding to finding REP interfaces of buses, bus-lines etc.

VI. Conclusion

The variety and heterogeneity of Internet of Things devices make it difficult to store their descriptions in relational databases as addition of each new device requires changes in the database structure. In this paper we presented an implementation of a Resource Directory based on the Sedna XML database, thus natively supporting storing of various resource descriptions. This native XML storage support, i.e. ability to directly process XML data without having to do XML serialization/deserialization first, results in shorter query execution time and increased flexibility in resource management.

Sedna is a free, open source native XML database with drivers for various set of languages. Functionality of the Java driver used in the implementation is a subset of the functions available in the corresponding C++ driver. One of the missing functions of the Java driver was setting a limit on the query result set response. During evaluation, a lot of resource descriptions were similar and queries could result in more than 800.000 descriptions, this like result set gathering should be limited in order to avoid database deadlocking. Limiting response in the XQuery does not solve problem in deep because internal calculation works with all matched resource description. XML is case sensitive as well as querying.

References

[1] Tsiatsis, V., Gluhak, A., Bauge, T., Montagut, F., Bernat, J., Bauer, M., Villalonga, C., Barnaghi, P., Krco, S. "*Real World Internet Architecture. In: Towards the Future Internet - Emerging Trends from European Research*". IOS Press, Amsterdam, April 2010.

[2] SENSEI, "*Integrating the Physical with the Digital World of the Network of the Future*", FP7 project, www.sensei-project.eu.

[3] RESTLET, "*Open source REST framework for the Java*", www.restlet.org.

[4] Srdjan Krco, Jelena Vuckovic, and Stevan Jokic, "*ecoBus – Mobile Environment Monitoring*", Towards a Service-Based Internet Third European Conference, ServiceWave 2010, Ghent, Belgium, December 13-15, 2010. Proceedings pp 189-191.

[5] Smart Santander project, http://www.smartsantander.eu/

[6] Extensible Markup Language (XML) www.w3.org/XML/

[7] Modis group, Sedna XML DBMS, http://modis.ispras.ru/Development/sedna.htm.

[8] W3C XML Query (XQuery), http://www.w3.org/XML/Query.

[9] Apache JMeter, http://jakarta.apache.org/jmeter/

[10] Restlet - RESTful web framework for Java, www.restlet.org/

Sadržaj

U ovom radu predstavljena je implementacija Direktorijuma Resursa zasnovana na XML bazi podataka i

izvršeno je poređenje performansi te implementacije sa implementacijom zasnovanoj na relacionoj SQL bazi podataka. Rezultati testiranja pokazuju da XML baza podataka pruža veću fleksibilnost u upravljanju resursima uz kraće vreme upita, dok se primenom relacione baze podataka ostvaruje kraće vreme u pristupu selektovanom resursu.

Evaluacija performansi Direktorijuma Resursa bazirana na XML BAZi PODATAKA

Stevan Jokić, Srđan Krčo, Jelena Vucković, Nenad Gligoric, Dejan Drajic