Service Choreography: Present and Future

Tanveer Ahmed, Abhishek Srivastava
Discipline of Computing Science
Indian Institute of Technology Indore
Indore, India
Email: {phd12120101, asrivastava}@iiti.ac.in

***Abstract*—Service oriented architecture is widely adopted, accepted and appreciated for both horizontal and vertical integration of enterprise applications. The success is hugely aided by web service composition, which is a temporal collaboration of independent and loosely-coupled web services to execute a business process at runtime. In the Future Internet, the present practice of composition, the most popular variant of which is service orchestration, is expected to face a lot of problems due to its inherent centralized orientation. As a result, service choreography is widely viewed as an ideal replacement candidate. However, achieving a decentralized collaboration of autonomous, '*non-aligned*' and loosely-coupled web services is a challenge. To systematically enact a web service choreography, the present infrastructure has to address several complications. In this paper, we present several atypical issues faced by service choreography and the technological advancements required for its enactment. Based on the proposed solution, we develop a prototype with *'stateless'* RESTful web services. The entire prototype is deployed in-house (within the Institute) on a virtualized platform.**

***Keywords*-Service Oriented Architecture, Service Choreography, Future Internet**

I. Introduction

Due to the proliferation of web services, the service oriented architecture is witnessing its peek. This architecture has always had an orientation towards loosely coupled, platform independent and autonomous web services. The web services combine dynamically at runtime and form a temporal collaboration, commonly referred to as web service composition, to execute either a business process or a data & compute intensive scientific application. However, the present composition practice, service orchestration, face several challenges in the context of fault-tolerance and being a single point of failure. This downside is due to its inherent centralized alignment. In this context, we believe service choreography is *'the'* solution.

As is evident today, the Internet is rapidly and continuously evolving towards the *Future Internet* [1]. One of the constituents of the Future Internet, Internet of Services, aims at making services tradeable, discoverable and marketable at runtime. To address this paradigm shift, the present standard of describing services (e.g. WSDL for SOAP based and WADL for RESTful), and the execution procedure needs a second thought. We argue the present implementation architecture and infrastructure can't execute services and achieve a service choreography at the same time. The reason for this line of questioning is outlined in a few following points:

1) Since service choreography is enacted through a description language, e.g. WS-CDL, how do an autonomous service interpret the description, the execution order, the elements specified in the XML schema etc.? The present implementation of web services is out of scope for this purpose.

2) *'Hypothetically'*, if the service can somehow perceive the syntax and the language, then they need a specialized *proxy* parser for different description standards proposed in literature (BPELChor, LetsDance etc), again not in the present protocol stack.

3) How do the present implementation of services manage the context information? e.g. if a single service is *'dancing'* with multiple services, then how should a service become self-aware about intiating the appropriate *choreography*. Moreover, consider 'RESTful' architecture, the present *stateless* protocol suite does not allow context or *resource* storage to be *'stateful'*, let alone self-aware.

4) As envisioned, the Future Internet will host services on wireless devices(e.g. PDAs, IoT sensors etc.), with severe battery constraints. Under these circumstances, the issue of reliability is inevitable. In service choreography, there is no centralized controller, therefore, in this context how to recuperate from a failed or a partially failed choregraphy.

5) In 2013, the Internet hosted more than 3.5 billion devices [1]. By 2020, the number is expected to rise to *'26 billion'*[1]. In this context, the network and congestion issue would be appalling. In this environment, what is the ideal medium to exchange data and parameters among services. Should the services share a memory or should they exchange messages directly? If they share a medium, then how to direct a 'non-aligned' service to read from and write to the storage? Preserving the storage location is another issue (point 3).

In this paper, we present a possible solution to the questions asked above. The technological approach is devised, keeping in mind the present infrastructure and resources. To demonstrate the viability in actual deployment, we have developed a web based prototype for achieving service choreography with '*RESTful*' services. The entire prototype is validated by executing a service choreography on a virtualizated platform (XENServer).

II. Proposed Model

The technological solution presented in this paper, relies on some of the existing works in literature, and software implementations available in the open source community. We lay these technological foundations as a stepping stone to present the solution. It should be pointed here that some works in literature, e.g. [3] [4] etc., propose addition of an extra layer or an extra interface on top of a service. To practically deploy such schemes, require significant changes in the existing implementation of web services. Our motivation is to remove the extra layer, use the existing infrastructure, and enact a service choreography. We discuss the proposed work in this section.

In the Future Internet, we envision all type of services, whether Human Provided Services or Software Provided Services (SOAP or RESTful), to become tradeable and executable. In this scenario, the ideal candidate for describing all the essential properties is the

[1] http://www.gartner.com/newsroom/id/2636073

'Unified Service Description Language' (USDL)[2]. USDL is a comprehensive suite, describing services from multiple perspectives, e.g. Pricing, Function, Service Bundle, SLA, Legal etc. In the proposed model, we rely on this language for service description. We also envision compute intensive scientific applications to become executable (decentrally) via service oriented architecture, deployed on a cloud based infrastructure. Therefore, to tackle the issues outlined previously, we root the technological groundings in 1) Cloud Computing 2) Distributed Federated Enterprise Service Bus (DSB) 3) Event Driven publish-subscibe architecture 4) Semantic Space.

In the last few years, cloud computing paradigm has emerged as a prospective candidate addressing a wide exhibit of computing necessities. In our opinion, the cloud, or the '*federation of clouds*' is the ideal backbone to enact a service choreography.

To answer the questions asked in points 1-3, and to cater to the requirements of a stable delivery platform, we rely on a 'Distributed Federated Enterprise Service Bus'(DSB). Though a prototype DSB is proposed in [2], but the implementation is not sufficient to enact a service choreography. The focus of [2] is to serve the needs of the Future Internet only. The prototype, as described in [2] uses Binding Component(BC) to plug executable services onto the DSB. We propose a few extensions. First, the BC must be extended with the functionality required to parse the context information described via multiple description languages. In the proposed work, this component also acted as the proxy parser. Second, the listener component (the default component for each service in the Middleware) of each service should allow context information (for a *resource* or storage location) to be stored locally. In this way, without adding an extra layer on the devices itself, services (even implementing the RESTful framework) can communicate and invoke each other independently. Due to space constraints, we don't discuss much. However, a comprehensive discussion on the Middleware is available in [2].

To address the congestion related issues raised above (point 5), one of the proposals, the proposed work rely on is: event driven architecture. The event driven publish-subscribe model has garnered a significant attention from both academia and industry, therefore reliance on such an architecture to exchange messages among services, is the ideal choice. The services participating in a service choreography should follow this architecture, as it allows a 'throttled' load on the underlying network, and also prevents a pointless power consumption, especially in the case of mobile devices.

To deliver a stable storage medium for parameter retention, traceability analysis etc. (point 5), a semantic space is utilized (At this point, it is understood that the listener component can maintain the storage location). Semantic space is a coalition of multiple technologies. The semantic data space is accessible via existing protocols, provide virtualization of heterogeneous data centers, and is deployable on cloud. Moreover, the storage itself is offered as a web service, thereby affirming to the standards of SOA. A state-of-the-art implementation is available in OpenLink Data Spaces[3]. Next, we address reliability (point 4), and the issue with the underlying Middleware.

It is a known fact that a 'distributed' ESB, e.g. Petals, deliver messages no matter if a service is available or not[4,5]. It should be noted here that the prototype proposed in [2], extended this open source implementation. The basic implemented architecture (of Petals) allow two types of deliveries: Fast and Reliable. We believe in real time business applications (in such scenarios it is understood that fast mode is not desirable, for obvious reasons), availability is an important criteria. For example, if one is executing a case-oriented workflow, e.g. StockQuotes during Peak Hours, without the centralized controller, then the 'reliable-mode' will wait for the temporarily unavailable service to be plugged in again. For time critical applications, this is problematic. Therefore, to address the issue of reliability, a 'workitem' must be instantiated with redundant services available for immediate execution. The services should be discovered well in advance, thereby removing the process of *re-inventing the wheel* (discovering service again from *'Hybrid'* registries). Although, the functionality of dynamic discovery can be added, but it is noteworthy that the availability of redundant services allows for the inherent capabilities of load balancing, dynamic adaptations and a reduced execution time. The selection of a service, in that case should be grounded on an inexpensive, lightweight and a greedy technique.

Lastly, recovery. As far as recovering from a failed or a partially failed choreography is concerned, there are two choices: 1) Rely on the services in the upper hierarchy to re-initiate the blocked process flow or 2) Depend on a third party, e.g. the DSB itself, to recover the choreography. To rely on an autonomous web service is a rather strong assumption. Therefore, reliance on the third party is the only choice. Recovering from a failed choreography makes room for further development.

III. PROTOTYPE DEVELOPMENT

A quick prototype, based on the proposed solution was developed. RESTful web services with listener and binding component was developed using JAVA programming language. The present implementation does not include a DSB and a true semantic space. Therefore, stubs were programmed for the two components, and only a distributed shared memory (DSM), e.g. MozartSpace[6], was implemented. The distributed shared memory itself was offered as a service, thereby the entire communication was based on the alliance of the service oriented architecture and the event driven architecture. The services were deployed on a Quad-Core Processor with 8 GB RAM, on a virtualized platform within the computing lab of the Institute.

REFERENCES

[1] Issarny, Valrie, Nikolaos Georgantas, Sara Hachem, Apostolos Zarras, Panos Vassiliadist, Marco Autili, Marco Aurlio Gerosa, and Amira Ben Hamida. "Service-oriented middleware for the Future Internet: state of the art and research directions." Journal of Internet Services and Applications 2, no. 1 (2011): 23-45.

[2] Baude, Franoise, Imen Filali, Fabrice Huet, Virginie Legrand, Elton Mathias, Philippe Merle, Cristian Ruz et al. "ESB federation for large-scale SOA" In Proceedings of the 2010 ACM Symposium on Applied Computing, pp. 2459-2466. ACM, 2010.

[3] Barker, Adam, Christopher D. Walton, and David Robertson. "Choreographing web services" Services Computing, IEEE Transactions on 2, no. 2 (2009): 152-166.

[4] Fernandez Hector, Cedric Tedeschi, Thierry Priol. "A Chemistry Inspired Workflow Management System for Decentralizing Workflow Execution", IEEE Transactions on Services Computing, doi: 10.1109/TSC.2013.27 (pre-print).

[2]http://www.internet-of-services.com/index.php?id=264

[3]http://virtuoso.openlinksw.com/

[4]https://doc.petalslink.com/display/petalsesb31/Petals+Enterprise+Service+Bus

[5]http://en.wikipedia.org/wiki/Enterprise_service_bus

[6]http://mozartspaces.org/