Peer-to-Peer Orchestration of Web Mashups

Abul Ahsan Md Mahmudul Haque, Department of Computer Science, University of Tromsø, Tromsø, Norway
Weihai Yu, Department of Computer Science, University of Tromsø, Tromsø, Norway

ABSTRACT

Web mashups are web applications built on top of external web services through their open APIs. As mashups are becoming increasingly complex, there is a need for systematic support for their development and orchestration. This paper presents a peer-to-peer approach to mashup orchestration where a network of agents carries out orchestration using continuation-passing messaging. The approach supports exception handling and recovery. Our experimental results show clear performance gains of the approach over traditional centralized orchestration in service-oriented computing and orchestration done by application servers hosting mashups.

Keywords: Composite Services, Continuation-Passing Messaging, Decentralized Orchestration, Exception Handling, Peer-To-Peer Network, Rollback

INTRODUCTION

The Web technology is under continuous evolution (Figure 1). In the beginning, every interaction between a user and a web application involves a round of HTTP messages between the web browser and the web server (1 in Figure 1). This results in long delays and large amount of network communication. With browser-side scripting (Flanagan, 2011), a considerable amount of interaction can stay at the browser side (2 in Figure 1).

Meanwhile, an ever-growing large number of web applications provide open services, as data and operations, through published APIs. New web applications, called mashups, provide new web applications or services by combining the functionality and data from open web services (Di Lorenzo et al., 2009; Endrulli et al., 2012; Van Acker et al., 2011; Yu et al., 2008). ProgrammableWeb (www.programmableweb.com), for instance, lists thousands of open services and mashup applications.

Successful adoption of mashups requires both development-time and run-time support (Yu et al., 2008). Development-time support includes the tools for correct service invocation through open APIs and for extraction and conversion of data obtained from the external services. Run-time support, known as orchestration of mashups, is the conduct of the execution of mashups.

DOI: 10.4018/ijaras.2014070103
Current mashup applications are developed or orchestrated at the servers hosting the mashup application, or even at the browsers (Van Acker et al., 2011). That is, the application servers or web browsers call external services and provide composed services. Programmers have to deal with low-level details like data extraction and conversion, execution order, exception handling, and so on. Furthermore, every single interaction between a mashup application and an open external service involves a round of HTTP messages between the application server hosting the mashup (or the web browser) and the web server of the external service (3 in Figure 1). An observation of the popular open services is that many operations return a list of data elements. These data elements are then processed in loops, which often further invoke the same or different open external services. These messages introduce long delays and large amount of network communication. It would therefore be preferable that a significant amount of interactions remain at or near the open external services (4 in Figure 1).

As web mashups are still in its infancy, there is great potential and need for further extension. In particular, a lot can be learned from service-oriented computing (SOC), with the more traditional ways of services composition and orchestration. Suppose that we would like to build a web application for a conference as a mashup. The application collects useful information for the conference attendees, such as literature references and sightseeing attractions, which can be obtained from external services like digital libraries. The application can even automatically build interest groups and set up extra discussion sessions or social events using external social network services. It may happen that due to unexpected reasons, certain partially executed or completed operations need to be rolled back. For example, due to time conflicts or unavailability of conference rooms, establishment of some groups or sessions must be undone. Currently, no mashup building tool is able to support all features this application needs, such as exception handling and rollback. Most of these features are already well supported in different SOC approaches.

SOC focuses on cost-effective construction and integration of sophisticated applications within and across organizational boundaries. Therefore unlike web mashups, service compositions generally limit themselves within enterprises or between enterprises with mutual agreements. SOC is typically built on workflow technology and provides more structural and feature-complete support than web mashups. (Yu et al., 2008) discusses the differences between mashup and SOC, with an emphasis on support for mashup application development.

Usually in SOC, dedicated central engines carry out the orchestration of composite ser-
Self-Adaptable Discovery and Composition of Services Based on the Semantic CompAA Approach


[www.igi-global.com/chapter/self-adaptable-discovery-composition-services/68949?camid=4v1a](www.igi-global.com/chapter/self-adaptable-discovery-composition-services/68949?camid=4v1a)