Chapter 12
The MACE Approach for Caching Mashups

Osama Al-Haj Hassan
University of Georgia, USA

Lakshmish Ramaswamy
University of Georgia, USA

John A. Miller
University of Georgia, USA

ABSTRACT
In recent years, Web 2.0 applications have experienced tremendous growth in popularity. Mashups are a key category of Web 2.0 applications, which empower end-users with a highly personalized mechanism to aggregate and manipulate data from multiple sources distributed across the Web. Surprisingly, there are few studies on the performance and scalability aspects of mashups. In this paper, the authors study caching-based approaches to improve efficiency and scalability of mashups platforms. This paper presents MACE, a caching framework specifically designed for mashups. MACE embodies three major technical contributions. First, the authors propose a mashup structure-aware indexing scheme that is used for locating cached data efficiently. Second, taxonomy awareness into the system is built and provides support for range queries to further improve caching effectiveness. Third, the authors design a dynamic cache placement technique that takes into consideration the benefits and costs of caching at various points within mashups workflows. This paper presents a set of experiments studying the effectiveness of the proposed mechanisms.

INTRODUCTION
Web 2.0 is drastically changing the landscape of the World Wide Web by empowering end-users with new tools for enhanced interaction and participation. Among Web 2.0 applications, mashups (Programmable Web, 2009) are becoming increasingly popular as they provide end-users with high degrees of personalization. Conceptually, mashups are Web services that are created by end-users who also consume their results. They offer high level of personalization because they are developed by end-users themselves as opposed to regular Web services which are designed by professional...
developers. (Throughout this paper, the term Web services refer to traditional Web service model in which a service provider creates and deploys Web services).

Mashups basically collect data from several data sources distributed across the Web, which would then be aggregated, processed, and filtered to generate output which would be sent to end-user. Several mashup platforms exist on the Web including Yahoo Pipes (Yahoo Inc., 2007) and Intel MashMaker (Intel Corp., 2007). The unique features of mashups, represented in high personalization and end-user participation, pose new scalability challenges. First, giving end-users the privilege of designing their own mashups causes a mashup platform to host a large volume of mashups which implies that the scalability requirement for mashup platforms is much higher when compared with Web services portals. Second, large volumes of mashups also imply that the opportunities for data reuse are minimal unless specialized mechanisms to boost data reuse are adopted. Third, mashups fetch data from several data sources across the Web; these data sources differ in their characteristics and their geographical distribution. Finally, mashups may be designed by non-technical-savvy end-users, and hence they are not necessarily optimized for performance. Unfortunately, scalability and performance challenges of mashups received little attention from the research community. Although there have been some studies on the performance of traditional orchestrated Web service processes (Chandrasekaran, 2003), to our best knowledge, no studies have investigated efficiency and scalability aspects of mashups or proposed techniques to tackle them.

This paper explores caching as a mechanism to alleviate the scalability challenges of mashups. Caching is a proven strategy to boost performance and scalability of Web applications. For example, Web content delivery and Web services have long adopted caching (Wang, 1999). Several caching techniques have been specifically developed for Web services (Tatemura, 2005; Terry, 2003). However, most of these techniques cannot be directly used for mashups because of some significant differences between Web services and mashups. We need a caching framework that not only takes into account the structural characteristics of mashups but is also adaptive to the various dynamics of the mashup platform.

**Contributions**

This paper describes the design and evaluation of MACE (mashup cache) - a server-side cache framework for the mashup domain. MACE is sensitive to the structural composition of the mashups, and it can store results at intermediate stages of mashup workflows. The design of the MACE framework embodies three original contributions.

- We design a mashup structure-aware scheme for indexing cached data which enables MACE to efficiently discover whether any of the currently cached data can be reused in the execution of a newly created mashup.
- We incorporate taxonomy-awareness and provide support for range queries to further increase reuse of cached data.
- We present a dynamic cache point selection scheme that estimates the benefits and costs of caching data at different stages of mashup trees. Our approach selects a set of points that collectively maximize the benefit-to-cost ratio of caching data at those points.

We have evaluated the MACE framework through several sets of experiments. Our experiments demonstrate that MACE significantly improves the scalability and efficiency of mashup platforms. In the following sections, we discuss our motivations for doing this research and describe our mashup model and architecture. We follow that by proposing our index structure that