Optimising Web Services with Intermediaries

Mark Nottingham <mnot@akamai.com>

Abstract

Intermediaries are often used to scale the Web infrastructure, offering improved performance, availability and scalability of services. This brief explores the possibilities of scaling Web Services (services exposed by XML messaging solutions like SOAP and XML Protocol) through optimisation in intermediaries, and proposes further work which leverages XML Protocol's features to help scale them and improve performance.

Introduction

As the Web has grown, there has been an increasing need to address issues such as the scalability (the ability to handle growth in load efficiently), performance (as measured by the end users' perceived latency) and availability of Web sites. Intermediary solutions to these problems have been used for some time; first as caching proxies, and later in Content Delivery Networks, which use surrogates (intermediaries which act on behalf of the server, rather than the client) dispersed widely over the network to provide service.

HTTP intermediaries rely on certain aspects of requests and responses flowing through them to offer these benefits. For example, caching works because requests can be easily identified (through the Request−URI and a few headers), and the responses associated with them can be reused according to the HTTP's cache coherence rules.

While Web Services use lower−layer intermediaries (such as HTTP proxies and surrogates or SMTP relays) for transit, those devices' optimization mechanisms aren't aware of the semantics of a Web Services message, and therefore can't fully exploit them.

For example, while it's possible to describe cacheability for a Web Service's underlying HTTP response (e.g., with a Cache−Control or Expires header), this is only a basic means of providing hints about the message. In the best case, an HTTP cache would be indexed on the entire request as a string, meaning that a trivial change in the request's XML would cause it to be indexed as a separate entity.

This means that, by default, only the simplest Web Services can leverage intermediaries for optimization in very limited ways.

To allow intermediaries to more fully optimise Web Services, such optimisation need to be standardized, just as cacheability information for lower−layer HTTP entities is standardized in the HTTP. This will allow intermediaries to optimise a broad range of Web Services without service−specific code being executed.

This paper outlines potential use cases for Web Service intermediary optimization, proposes a framework for optimization, and identifies potential optimization techniques.

Optimisation Use Cases

Although tentative, these use cases help illustrate the potential of optimizing Web Service intermediaries.

StockQuote Service

By caching response elements containing rapidly−changing financial data for a period, a Stock Quote Service could offer enhanced end−user perceived performance whilst reducing load on centralized servers. Furthermore, slowly−changing data in the same response could be given separate, longer−term cacheability, with different criteria for invalidation from the cache.

News Service

An XML−based news 'channel' Service can take advantage of a regular publication schedule to cache article summaries with an absolute time until validation (with the possibility of using partial content updates). Additionally, because some servers may provide many such services, channel requests and responses may be aggregated into a single message interchange for efficiency.

Distributed Authentication Service

A centralized Web site user authentication Service can exploit geographic locality in client behaviour by allowing a distributed group of caches to keep authentication state, rendered as XML, at the 'edge' of the network. If the user changes their password, the request for change can act as a trigger for invalidating the cached entry.

File Store Service

An 'Internet hard drive' Service, where users write to and read from a Service as if it were a network−available disk, could be distributed to a number of 'edge' servers to improve end−user perceived latency by exploiting locality in their access patterns. This could be achieved through a combination of store−and−forward into a cache (i.e., write caching), reading from the cache, and invalidation events to stimulate synchronisation with a centralized server.

Order Queue Service

With store−and−forward techniques, service intermediaries can provide higher availability for a service than a centralized server alone, whilst offering the potential to manage load on the central server by aggregating messages to it.Voting, Poll and Auction Services

'Interactive' Services can take advantage of 'best−guess' information in cache whilst updating critical information through message triggers and element invalidations.

Further Work

This brief attemps to motivate research regarding optimising mechanisms in service intermediaries. Hopefully, this will generate interest in standardization of such techniques, development of a framework for their use, and integration into Web Service toolkits and products.