Adaptively Caching Distributed Components

Christoph Pohl [*]

Abstract

Caching is commonly employed to achieve locality of reference in distributed systems. It is however hardly supported by middleware platforms like EJB or CCM, despite being a non-functional aspect. This forces application programmers to explicitly use design patterns etc. Our approach is to *transparently cache component attributes* while adapting to changing access characteristics, thus fully preserving distribution transparency.

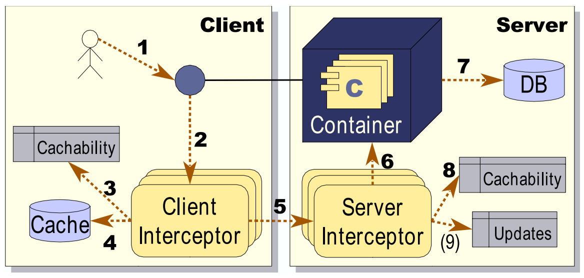
Typical use cases comprise highly interactive "fat client" applications operating on server-side data models, e.g. e-learning or distributed multimedia, but also web servers accessing components on application servers.

Approach

Our concept is implemented as an extension to the open-source J2EE application server JBoss[1]. It currently follows the EJB component model, hence attributes are exposed on component interfaces by means of `get/setXyz` method pairs. Cacheability of attributes is preconfigured at design time using tagged JavaDoc comments on these methods, which are evaluated by an XDoclet[2] template to generate descriptors etc.

JBoss features a dynamic proxy architecture for component stubs based on the Java Reflection API, including an interface for chained interceptors similar to those defined by the OMG. We implemented an additional pair of configurable client-/server-side caching interceptors as shown in the figure.

*TU Dresden, Institute for System Architecture, Chair for Computer Networks – http://www.inf.tu-dresden.de/~cp6



The client interceptor retains results of cacheable attribute queries in a local in-memory cache like JCS[3]. Invalidation, update propagation, and adaptation to changing read/write ratios is accomplished via additional payload contexts attached to regular invocations by the interceptors in a piggy-back manner.

Outlook

Potential performance benefits have been proved by first tests. In-depth measurements will follow as soon as the implementation matures. Further investigations will include *prefetching*, i.e. possibilities to transfer data to client-side caches *before* it is queried; automatic generation of *value objects*, i.e. grouped of attributes to decrease the transfer overhead; and persistent caching to permit off-line client scenarios. The results of this work are integrated into the COMQUAD project[4].

Notes

[1]http://www.jboss.org/
[2]http://xdoclet.sf.net/
[3]http://jakarta.apache.org/turbine/jcs/
[4]http://www.comquad.org/