COMPOSING REPLICATION SERVICES

SERVICIOS DE REPRODUCCIÓN COMPUESTA

Stéphane Drapeau¹, Claudia Lucia Roncancio¹
stephane.drapeau@imag.fr / claudia.roncancio@imag.fr

Recibido: noviembre 12, 2003 / Aceptado: enero 30, 2004 / Publicado: febrero 18, 2004

ABSTRACT. The RS2.7 Replication Framework revisits the replication function in order to provide component-based services for several kind of environments. It clearly identifies what minimal functions are relevant to replication: binding replicas and synchronizing them in order to support the right levels of coherency. This paper presents how the support of the coherency level can be decomposed with respect to two dimensions: functional and scheduling. Playing with these two dimensions allows to provide different replication services by merely assembling RS2.7 components. A prototype of RS2.7 is operational and has been applied to a platform for interactive networked applications.

KEYWORDS. Replication, consistency, coherency, adaptable framework. Distributed computing, RS2.7, replication services.

RESUMEN. El marco de reproducción RS2.7 visita de nuevo la función de reproducción para poder proporcionar servicios basados en componentes para entornos de diversos tipos. Identifica claramente qué funciones mínimas son relevantes en la reproducción, al ligar y sincronizar réplicas para poder soportar los niveles correctos de coherencia. El presente artículo aborda cómo el soporte para el nivel de coherencia puede ser descompuesto con relación a dos dimensiones: funcional y programada. Jugar con estas dos dimensiones permite suministrar diversos servicios de reproducción con el simple ensamblaje de componentes RS2.7. De esta forma, se aplicó un prototipo de RS2.7 operacional a una plataforma para aplicaciones interactivas de red.

PALABRAS CLAVE. Computación distribuida, marco de reproducción RS2.7, marcos adaptables, servicios de reproducción.

Introduction

Distributed computing requires mechanisms to ensure system availability and reliability with different purposes such as fault-tolerance and performance scalability (e.g., parallel computing). Replication is at the heart of solutions for all the preceding issues and is usually declined with many different flavours. Unfortunately, apart from the principles involved, used replication protocols have little in common. They generally implement some dedicated ad hoc replication support. According to system designer objectives, a replication protocol with particular constrains must be used. Defining such a protocol could be easier with an adaptable support. Besides, an adaptable support may allow the system designer to try different protocols comparing their impact/performance on data. This approach introduces new principles for the database management systems.

This paper gives an overview of the RS2.7 Replication Framework (for a complete presentation see [3]). Our approach revisits the replication function in order to provide services for several kind of environments. In the design of RS2.7 we have considered two conditions to be fulfilled in order to provide adaptability. The first one is the separation of concerns which means that the functional scope of the framework is only devoted to replication (Section 2.1). The second condition is the decomposition of the coherency support into a large number of components, each one playing a particular role. This support can be decomposed with respect to two dimensions: scheduling (Section 2.2) and functional (Section 2.3). We propose a fine grain

¹ Lab. LSR-IMAG, BP72, 38402 Saint Martin d’Hères, France -www.lsr.imag.fr
decomposition allowing to tune replicas coherency simply by assembling the most optimal implementation (Section 3). Section 4 is devoted to related work. Our conclusions and future work are given in Section 5.

RS2.7: an adaptable replication framework
Definition and scope of RS2.7

Adaptation can be obtained through parametrization. However, it seems quite impossible to provide a general purpose replication protocol that can be parameterized to accommodate all runtime environments. Indeed, we notice that large amounts of code need to be replaced from one case to another. Further, increasing the number of parameters tends to heavily increase the code complexity. We argue for a component-based (framework) approach where pieces of code (i.e., components) can be changed for adaptation purposes. A replication service is an instance of the framework.

Separation of concerns is a key issue to adaptation. In that respect, we do not intend to provide a full-fledged framework in terms of replication functions because this could reduce reusability. Thus, a replication service focuses on two features: life cycle management of groups of replicas (e.g. their creation and deletion) and inter-replicas synchronization protocols (named coherency protocols in the following). The replication time, the replication degree and the replicas placement depend on the purpose of replication (e.g. load balancing or fault tolerance). These aspects are in charge of the systems/applications using the replication service. Requirements of these applications lead to different replication services (RS2.7 instances).

We adopt the notion of coherency model which defines how users perceive the different replicas of an object. We classify coherency protocols according to four models. In the one copy equivalence model, replicas are always equivalent. The divergent replicas model allows replicas divergence. It is possible to characterize this divergence by giving some guarantees on R/W operations and their execution order. The convergent replicas model allows replicas to diverge but they eventually converge at some point. A limited level of inconsistency based on conditions like delay, periodic, time points, version, numerical, object or event conditions is supported. We distinguish two kinds of models: the convergent replicas models with reads on divergent replicas where each replicated object has an owner where updates are first applied, and the convergent replicas models with writes on divergent replicas where reads and writes may be done on not up-to-date replicas.

Various requirements concerning concurrency control (CC) or fault tolerance (FT) must to be considered when implementing specific coherency protocols. Thus, replication, CC and FT are not independent even though separating them and clearly defining their interactions is a way to enhance adaptability and reusability.
Application execution requires a consistency model which is a more or less formal specification of how the memory appears to an application. Consistency models are implemented by consistency protocols that manage objects taking into account replication, CC and FT. With our proposed coherency models, consistency protocols do not have to deal with replication issues. The objective is to hide issues related to replication behind coherency models (Figure 1). RS2.7 implements the appropriate coherency protocols. This approach promotes the use of RS2.7 in several contexts supporting transactions [7], fault tolerance, distributed shared memory (DSM), etc.

2.2 Structural decomposition

For adaptability purposes, RS2.7 must support several coherency protocols. Nevertheless, the large variety of existing coherency protocols complexifies the design of the framework. In order to be able to provide a generic replication framework, adaptability is introduced by providing abstractions for defining coherency protocols. These abstractions consist of five generic phases: the access, coordination, execution, validation and response phases. The differences between protocols are characterized by the approach they use in each phase and the order the phases are executed. The abstract coherency protocol is the minimal shared part between all existing protocols.

For each phase, we define a component with a generic interface. Thus, it is possible to change some specific phases of a particular coherency protocol to obtain new ones.

2.3 Functional decomposition

In order to obtain adaptability inside the framework, functional decomposition is in turn applied to coherency protocols. This means that common functionalities have been extracted from protocols. We propose a functional architecture distinguishing the components involved in the construction of coherency
protocols [2]. Each component has an interface covering a particular function that can be implemented in several ways. With the RS27 component approach, protocols can be built by assembling components for applications with particular requirements. Moreover, existing protocols can be easily enhanced by incrementally adding new components to existing assemblies.

The functional architecture provides four categories of components. Kernel components are considered as the basic level to construct simple protocols. These components mainly concern replica life-cycle, their communication and interactions with the user application. Components common to all coherency models are those related to general synchronization issues. Components dependent on the coherency model and components dependent on the coherency protocol add particular model/protocol dependent components like replica's role, conflict detection/resolution, synchronisation group.

Implementation and experimentation

The basic architecture principle that governs RS2.7 consists in interposing mediation objects to access replicas in order to provide transparency. They are linked to binding objects that bind replicas between each other. The mediation objects also manage replication by implementing the appropriate coherency protocol.

The coherency protocol that manages a replicable object is distributed on each replica through the binding representatives. The construction of a particular protocol consists in assembling the appropriate functional components (Section 2.3) within each phase of the abstract protocol (Section 2.2), and in integrating these different phases (scheduling components) through the relevant scheduling algorithm. All this is embedded into binding objects.

In order to validate the decomposition of the replication functionality proposed in Sections 2.2 and 2.3, several protocols implementing the four coherency models (Section 2.1), have been developed in Java. For the time being, the composition of these components and the composition of the phases are hand coded. Our first goal is to validate functional and scheduling decompositions.

A first kind of adaptability in the protocols is obtained by the fact that each component of the functional architecture has a generic interface allowing various implementations. For instance, the component update log component can be based on a log or a file or merely by directly performing the access to the replica state. In the same way, conflict detection and conflict resolution can be typically implemented in several ways according to the context. Adaptability can also be obtained by modifying a phase (adding, deleting components as well as changing the composition of the basic components) or by changing a binding representative.

A utilization of the replication framework has been done in the PING (Platform for Interactive Networked Games) European project [4]. The PING project intends to specify, develop and demonstrate a flexible and scalable architecture for large-scale interactive multi-participant applications over the Internet. The context is not transactional. In this implementation (developed in Java), the focus has been put on the separation of concerns among coherency protocols, concurrency control and consistency protocols as described in Section 2.1.
Related Work

Existing work may be classified into two categories: those proposing a support of replication protocols to achieve fault-tolerance and those proposing a more generic approach. In the first category, the range of provided replication techniques is limited to strong coherency among replicas [9,6]. Replicas are managed using group communication, implemented through multicast primitives. We find a similar approach in works around Corba [10,5].

Systems of the second category allow a large number of replication techniques [8,1]. However, these works propose more than a replication framework. They combine concurrency, replication and consistency protocols. They do not really isolate what is specific to replication and do not extract general replication functionalities. This limits adaptability, flexibility and reusability of their frameworks. For a deep analysis on works on replication see [2].

Conclusions and perspectivas

This work contributes to a clear separation of the replication functionalities in order to enhance adaptability in replication services. The RS2.7 replication framework provides support for replica life cycle management and for coherency protocols. Separation between coherency models and consistency models allows the use of our framework in several contexts like transactional or fault-tolerant ones. Moreover flexibility to support various coherency protocols is guaranteed by a scheduling and functional architecture that identifies several components involved in the construction of coherency protocols.

A first implementation of RS2.7 has been made in order to validate functional and scheduling decompositions [3]. A second prototype has been integrated in the Platform for Interactive Networked Games of the PING IST project. This experience was focused on the separation of concerns between coherency protocols, concurrency control and consistency protocols.

Building replication services is a difficult task. On-going work includes the use of the ObjectWeb Fractal component framework (http://www.objectweb.org/architecture/component/index.html) that allows to describe the composition and to generate an optimized implementation. This approach allows to take advantage of the openness of our framework and of its internal architecture. Composition may be static or dynamic, depending on trade-offs to be made between performance and dynamic adaptation (i.e., reconfiguration). In this respect, we intend to experiment component merging patterns for performance enhancement.

Acknowledgements. A grand merci to Patricia Serrano Alvarado and NODS members for their useful discussions on our work.

References


