A Generic and Flexible Model for Replica Consistency Management

Corina Ferdean and Mesaac Makpangou

INRIA Rocquencourt, France, BP 105, 78153 Le Chesnay Cedex

Abstract. This paper presents a flexible consistency model, aggregating a parameterized representation common for all the models along the spectrum delimited by strong consistency and eventual consistency. A specific model, required by a particular Data Object, is derived from this representation by selecting and combining the proper consistency parameters values.

1 Introduction

The basic unit of replication and of consistency management, that we consider, is the Data Object. A Data Object is basically a passive entity encapsulating any data -representing the object’s state- and the operations for consulting and manipulating that data -representing the object’s access interface-.

The problem we address consists in accommodating the suitable consistency model for a particular Data Object, when it is replicated, with its replicas being accessed concurrently. The base hypothesis that we consider are the variety of Data Objects and the heterogeneity, which happens in most cases, for the services provided by the same Data Object. Although the existence of a spectrum of models delimited by strong consistency and eventual consistency has already been identified [9], existing approaches remain too rigid with respect to the consistency level they capture along this spectrum. For example, bounding discrepancy observed when reading data and relaxing total ordering of concurrent updates could be both required by the same Data Object. However, at our knowledge, there is no existing consistency framework which support this combination. Also, not all update operation calls should be associated the same propagation policy. Neither this particular flexibility feature is met throughout the state of art on the consistency models.

2 Our Model Description

In order to meet various application needs, we define our model in two successive steps. They provide, respectively, the model’s genericity and flexibility features. The first step consists in identifying the different concerns of the consistency aspect. We attach to each concern one or several parameters. The second step consists in providing one or several options for each parameter.
gid, gid₁, gid₂ identifying groups of accesses
consistency_concern = liveness | safety
liveness = visibility | observed_state_quality
safety = pre-scheduling | scheduling

consistency_model = {consistency_constraint}
consistency_concern = visibility_constraint | observed_state_quality_constraint |
pre-scheduling_constraint | scheduling_constraint
visibility_constraint = ([gid]; transfer_instant)
observed_state_quality_constraint = ([gid]; tolerated_divergence)
pre-scheduling_constraint = ([gid]; execution_mode)>
scheduling_constraint = ([gid₁, gid₂]; scheduling_relation)

Fig. 1. Generic constraints on consistency concerns

2.1 Consistency Granularity

We define an access as an operation invocation, issued at a particular replica. We define a group_of_accesses as a collection of accesses, related by some common attribute(s) (e.g. the operation identifier, the caller identifier, the replica identifier). We define a special group named interface_group, which contains all the calls to all operations. According to its type, a consistency option can be associated to a group_of_accesses (by default this is the interface_group) or to a pair of two groups_of_accesses.

2.2 Generic Consistency Constraints

We classify the consistency concerns hierarchically on two levels. On the first level, we distinguish between liveness and safety concerns (Fig. 1) . A liveness constraint enforces the progression of replicas towards an equivalent state. It states that all operations issued at a replica should be transferred sooner or later to all the peers. A safety constraint enforces replicas convergence by correct scheduling of the global set of operations, issued at different replicas and which have to be applied all over. Scheduling aims to stabilize updates, by finding their position within the global history. This is done with respect to conflicting and/or non-commutative updates issued concurrently at peers. On the second level, we refine a liveness concern into a visibility and an observed_state_quality concern, and a safety concern into a pre-scheduling concern and a scheduling concern. A visibility_constraint targets the progression of peers towards the final state, by enforcing the spreading of locally issued updates. An observed_state_quality_constraint targets the progression of the local replica towards a particularly advanced state (in particular, the final state), by requiring relevant remote updates to be pulled locally.

A visibility concern is parametrized by the transfer_instant. It specifies when the spreading of local updates should be proceeded. An observed_state_quality concern is attached the tolerated_divergence parameter. It specifies if a discrep-