Foundations of Speculative Distributed Computing
(Invited Lecture Extended Abstract)

Rachid Guerraoui

EPFL, Switzerland

The Pitch

Speculation is frequent in distributed computing. It is even the norm in distributed algorithms that are designed for practical purposes. Yet, speculation is difficult and it has, so far, led to algorithms that are intricate and hard to reason about, let aside prove and test. The reason is simple: speculation involves different execution paths that are usually intermingled in the same algorithm, adding to the difficulties of concurrency and communication, inherent to general-purpose distributed computing.

This talk suggests the possibility of a principled approach to speculation and, indirectly, of a well-founded approach to the design and implementation of practical distributed algorithms.

Speculation in a Nutshell

Speculation is the idea that the design of an algorithm follows an *if-then-else* control structure, obeying the celebrated political motto: *hope for peace but plan for war*.

In short, under certain good conditions, i.e., the *speculation*, a specific path of the algorithm, i.e., the *speculative path*, is executed. When the conditions are not met, i.e. in the *bad case*, a *backup path* is executed instead. The rationale behind this approach is twofold: *(i)* the probability for the speculation to hold is high and, *(ii)* the complexity of the speculative is significantly lower than that of the backup path.

Examples of speculative algorithms are numerous. They include cache coherence protocols, multi-thread assignment techniques, Ethernet congestion control mechanisms, transactional memory systems, as well as mutual exclusion, lock-free, replication and agreement algorithms.

The Many Faces of Speculation

Speculative algorithms can be classified according to at least three categories.

One form of speculation, arguably the most common, is the *direct* one with exactly one speculative and one backup paths. This is for instance the case with *fast* mutual exclusion algorithms where, in the absence of contention, the number of steps needed to enter a critical section is significantly lower than the one needed in the presence of contention.

N.A. Lynch and A.A. Shvartsman (Eds.): DISC 2010, LNCS 6343, pp. 204–205, 2010.

A more sophisticated form of speculation is the *nested* one where the backup path might itself be composed of a speculative path and another, nested, backup path. An example of such a form of speculation is replication algorithms that behave in an efficient manner, say with constant complexity if the system is synchronous and prone to failure, and finally into a backup path with potentially undefined complexity if the system is not synchronous. One could also consider different types of contention and devise mutual exclusion algorithms that degrades gracefully according to the type of contention encountered, e.g. total contention, interval contention, step contention, etc. Early deciding algorithms, where the complexity depends on the actual number of failures that occur in an execution can also be viewed as nested speculative algorithms.

An even more sophisticated form of speculation is the *recursive* one where the distributed algorithm might keep switching back and forth between the speculative and the backup paths. This is typically the case with eventually synchronous agreement algorithms, which keep looping over a speculative path that terminates whenever the system becomes synchronous. Interestingly, speculation is used here to ensure termination, i.e. to turn the complexity from infinite to finite.

Speculative Principles

Like any distributed algorithm, a speculative algorithm A is supposed to solve a problem P assuming a set of executions (or traces) M, commonly called a model. The very characteristic of a speculative algorithm is that, under M', a strict subset of M, the complexity of A is supposed to be lower than its complexity in M.

Part of the challenge underlying speculation is that, while A is supposed to perform efficiently under M', A is still expected to solve P under M. Hence the design of A requires to *detect* whether the speculation turns out to be true, and if it does not, to *safely* fall back to the alternative path.

Setting the grounds for a theory of speculation goes through addressing many challenging questions. Given a problem P and an algorithm A' solving P in a model M', is it possible to automatically transform A' into the speculative part of a more general algorithm A that solves P within a strict superset of M? Is there a refined speculation of P of which solutions can be viewed as speculative, composable, parts of the same speculative algorithm solving P. Are there, for any sub-model M' of a more general model M, a general *detector* that establishes whether a given execution path of M belongs M'.

Ideally, a theory of speculation would lead to devise the distributed counterpart of the *if-then-else* control structure where *(a)* the speculation, *(b)* the speculative algorithm and *(c)*, its backup, would be, first class, separate citizens of a distributed computation.