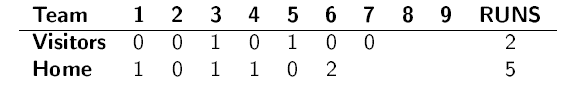
Abstract

Sistemi di storage moderni replicano i dati su più macchine al fine di garantirne persistenza, low latency e tolleranza ai guasti. La presenza di più repliche di un medesimo dato genera quindi problemi di coerenza fra i dati stessi. Tale coerenza può essere garantita mediante diversi modelli, ognuno con i propri punti di forza e le proprie debolezze. Chiaramente, non ne esiste uno valido in generale e la scelta deve essere fatta tenendo in considerazione quali proprietà del sistema riteniamo maggiormente rilevanti ai nostri scopi. Pertanto, prendendo in esame i due sistemi di key-value store distribuiti Amazon Dynamo e COPS, è stato possibile verificare, anche tramite confronti, come questi compromessi vengono raggiunti.

Introduzione

Ogni applicazione necessita di un livello di coerenza che cambia a seconda degli scopi che vuole raggiungere e delle priorità che vuole ottenere. Infatti, nel 2000 è stato congetturato, e successivamente dimostrato, un risultato teorico che prova che in ogni sistema distribuito non è possibile soddisfare contemporaneamente tutte e tre le seguenti proprietà: *availability*, ossia ogni richiesta riceve sempre una risposta su ciò che è riuscito o fallito non rimanendo mai in attesa indefinitamente, *partition-tolerance*, ossia il sistema continua a funzionare nonostante arbitrarie perdite di messaggi, *consistency*, ossia tutti i nodi vedono gli stessi dati simultaneamente. Questa conclusione è stata dimostrata nel 2002, dopo quasi trenta anni di ricerca, e va sotto il nome di *Teorema CAP*.

Per descrivere alcuni differenti modelli di coerenza, utilizzeremo come esempio i punteggi di una partita di baseball, memorizzati all’interno di un key-value store distribuito, come mostrato di seguito:



**Strong consistency.**  La garanzia più alta che possiamo raggiungere fornisce ad ogni client che effettua operazioni di lettura sempre l’ultimo valore aggiornato. Per implementare tale livello, è necessario un alto livello di sincronizzazione tra i vari nodi, che per essere raggiunto esige attese, causando un calo delle performance e la partecipazione attiva di tutti i nodi. In riferimento all’esempio, è possibile che ci venga ritornato unicamente il punteggio 2-5.

**Eventual Consistency.** Formalmente l’eventual consistency consente di ritornare un qualunque valore che è stato scritto su un dato che il client vuole leggere. In pratica, quello che questo livello garantisce è che se non vengono eseguiti nuovi aggiornamenti su un oggetto, eventualmente tutti gli accessi a quell’oggetto ritorneranno l’ultimo valore aggiornato.

I modelli di coerenza descritti sottostante, sono stati progettati per garantire un giusto bilanciamento tra le proprietà del teorema CAP: consistency, availability e partition-tolerance. Infatti questi modelli vengono situati al centro fra la Strong Consistency ed Eventual Consistency, come mostra la figura X.

proprio perché la “forza” di un modello di coerenza non è stabilita da quando o come propaga i nuovi valori tra le repliche, ma è definita dall’insieme dei possibili risultati che possono restituire.

Strong

Consistency

Eventual

Consistency

Intermediate

Consistency

**Consistent Prefix.** Questo modello garantisce che i dati ricevuti siano ritornati in sequenza (ordinati dalla replica master) delle scritture effettuate, in altre parole sono istantanee del sistema intero.

**Bounded Staleness.** Assicura che i risultati ricevuti non siano scaduti, infatti la “Staleness” è definita come il tempo di vita dei dati “T”. Il sistema garantisce che l’operazione di lettura ritorni un qualsiasi valore scritto nei T minuti passati oppure valori recentemente scritti.

**Monotonic Reads.**

**Read My Writes.** E’ l’unico modello che ha due comportamenti differenti in base al client che effettua l’operazione di lettura, ovvero si ha coerenza forte se è l’unico ad effettuare le scritture, altrimenti ritorna gli stessi valori dell’eventual consistency.

In definitiva, abbiamo stabilito che non esiste un modello di coerenza perfetto, che soddisfi le necessità di ogni