# Reading Report: Terry13

**Ricard Abril**

March 24, 2019

## 1 Summary

En aquet text, es parla de sistemes de emmagatzemament replicats, i ens parla de diferents models per guardar dades en els diferents sistemes de cloud, com AWS o Azure, i com afecta la aplicació d'aquets diferents models a la consistència de les dades com eventually consistent, i strong consistency. En el cas de la primera, de aplicar-la obtindrem un rendiment major, encara que es possible que certs usuaris en un moment donat llegeixin un valor que ja ha canviat. En el cas de la segona eliminem la possibilitat de llegir dades no actualitzades o equivocades, però el aplicar aquet sistema augmenta considerablement la latència.

**Read consistency Guarantees.**

En aquesta secció, s'introdueixen els diferents models de replicació de dades que s'explicaran en aquet text, aquestes es basen en el tipus de operació a executar per el usuari (Read o Write), i explica que una operació de Write es qualsevol que modifiqui part de un objecte de dades, i una de Read, qualsevol que nomes pretengui obtenir el valor de un objecte de dades.

**Strong consistency.**

Aquet model de consistència, granteix que una operació de Read, retornarà les ultimes dades escrites en un objecte en qualsevol cas.
En cas de que aquesta de operació de Read, també modifiqui el objecte de dades, el Read retornarà el objecte de dades amb les modificacions aplicades.

**Eventual consistency.**

Aquet model, es el que aporta unes garanties més febles. Per la qual cosa, si ha una de les repliques de les dades se li deixa de aplicar un Write i en aquell moment es llegida per un usuari, pot obtenir qualsevol de les dades passades de aquell objecte de dades.

**Consistent prefix.**

En aquet model de emmagatzemament de dades, el usuari te la garantia de poder veure una seqüencia ordenada de writes al objecte de dades en qüestió, fins al últim write realitzat.
Això ens aportarà beneficis, quan llegim multiples objectes de dades, o quan les nostres operacions de Write, actualitzen de forma incremental un objecte de dades.

**Bounded staleness.**

Aquet model, ens ofereix la garantia de que un Read, ens retornarà qualsevol dels valors que objecte de dades haguí tingut en un temps límit T, es a dir ens assegura que les dades no seran mai més antigues que T.

**Monotonic reads.**

També es denomina garantia de sessió, i es una propietat que s'aplica a un certs clients, aplicant aquet model, podrem fer lectures a dades obsoletes arbitràriament, igual que en la consistència eventual, però aquet model ens assegura que un segon Read, retornarà el mateix o unes dades més noves.

**Read my writes.**

Es una propietat que també saplica unicament a una sequencia de operacions realitzades per un unic client, i garantitza que totes les escriptures realitzades per el client, puguin ser llegides posteriorment per aquet.
Si un altre client modifica un objecte de dades, de totes formes el primer seguira lleguin la ultima dada que ell ha escrit.

**Baseball example.**

Al final d'aquet article, s'utilitza el baseball com a exemple de que no tots els integrants en un partit de baseball (Official scorekeeper, Umpire, Radio reporter, Sportswriter, Statistician, Stat watcher), necessiten el mateix tipus de garanties, per la qual cosa, cada un tindrà un model òptim ha aplicar.
Abans de explicar quin es el model idoni per a cada integrant del partit, explica com funciona un partit de Baseball, i també mostra algunes figures amb el codi del programa per "Seguir" el partit i com es veurien les dades del marcador aplicant diferents models.
Per els diferents integrants:

**Official scorekeeper.**

Aquet és el responsable de comptabilitzar les carreres dels equips i per tant de la puntuació del partit, per tant no pot permetre una lectura de una dada no consistent, ja que podria afegir o restar punts a un equip, però per altra banda tampoc necessita una lectura plenament consistent, ja que serà el únic que actualitzarà la puntuació. Per tant serà suficient amb un model Read my writes.

**Umpire.**

El àrbitre, si que necessitarà lectures consistents solides, ja que en Baseball, en algunes ocasions, es pot donar per guanyador a un equip si amb les carreres restants, es impossible que el rival pugui guanyar, de forma que si el àrbitre realitza una lectura errònia, podria donar per guanyador un equip de forma incorrecta. Per tant aquet necessita un model de consistència forta.

**Radio reporter.**

Els reporters de radio, segons explica al article, a USA, es habitual que aportin informació periòdica sobre els resultats de Baseball dels jocs en curs, per la qual cosa necessiten certa consistència en les dades, però degut a que aquesta informació es dona cada cert període de temps, no es necessària una consistència forta, sinó que el que es necessari es la garantia de cada X període de temps les dades seran més noves que les anteriors, per la qual cosa per aquet participant serà suficient amb un model de Bounded staleness.

**Sportswriter.**

El periodista esportiu, molt probablement no necessita de cap mena de consistència, ja que ell començarà a escriure la noticia sobre el partit un cop aquet, ja ha acabat per la qual cosa un model de consistència eventual, hauria de ser suficient, però la única forma de estar 100

**Statistician.**

El estadístic, és el encarregat de portar les estadístiques de carreres, outs i demes dades, tant del equip en general, com per cada jugador individualment, per tant necessitarà una consistència forta, ja que cada cop que les dades canviïn aquet les actualitzarà per a poder tenir estadístiques valides.

**Stat watcher.**

La gent que consulta les estadístiques dels seus equips i jugadors, tenen suficient amb una consistència eventual, ja que aquestes dades s'actualitzen un cop al dia, i si no son del tot recents, tampoc passa res.

**Conclusion.**

Al final d'aquet article, s'utilitza el baseball com a exemple de que no tots els integrants en un partit de baseball (Official scorekeeper, Umpire, Radio reporter, Sportswriter, Statistician, Stat watcher), necessiten el mateix tipus de garanties, per la qual cosa, cada un tindrà un model òptim ha aplicar. Abans de explicar quin es el model idoni per a cada integrant del partit, explica com funciona un partit de Baseball, i també mostra algunes figures amb el codi del programa per "Seguir" el partit i com es veurien les dades del marcador aplicant diferents models. Per els diferents integrants: En conclusió, els participants en el partit de Baseball, podríem pensar que son diferents tipus de aplicacions que necessiten accedir a certes dades en el nostre sistema, que tenen diferents, i per tant necessiten garanties també diferents.

**All of the six presented consistency guarantees are useful.**

Cada un del models, aporta unes garanties diferents, que seran útils per diferents aplicacions.

**Different clients may want differ-ent consistencies even when accessing the same data.**

En algunes ocasions, dos aplicacions volen llegir les mateixes dades, però encara i així, al no tenir les mateixes necessitats no necessiten les mateixes garanties, com per exemple el reporter esportiu i el comentarista de radio.

**Clients should be able to choose their desired consistency.**

Per el mateix motiu, cada client hauria de poder triar el nivell de consistència que més li afavoreix, ja que no tots necessiten cenyir-se a un model de consistència forta, i per altra banda en alguns casos no els es suficient amb un model de consistència eventual.

# 2 Assessment

El text explica de forma clara els diferents models de consistència de dades, i els seus punts febles i forts, cosa que pot ser útil saber per aplicar-la en un futur a diferents aplicacions.
En un primer moment, no sembla tenir massa importància el model de consistència a aplicar, ja que sembla lògic en un primer moment aplicar pràcticament sempre consistència forta, però es interesant poder veure que no en tot cas necessitem totes les garanties que ofereix la consistència forta i que es possible aplicar nomes les garanties necessàries per tal de no perdre tant rendiment.
Per tant si que recomanaria la lectura per el següent curs.