Le future d'Apache Cassandra™

DuyHai DOAN

Consultant freelance Java/Cassandra-Datastax/AWS/<rayer-la-mention-inutile>
Ex-évangéliste Apache Cassandra™

Ordre du jour

- La roadmap pour Apache Cassandra™ 4.0
- La fragmentation de l'écosystème
- La menace du Cloud
- L'éternel débat OSS vs Propriétaire

La roadmap 4.0

Petit rappel historique des releases

- 03/09/2013: Cassandra 2.0 (Lightweight transactions)
- 16/09/2014: Cassandra 2.1 (UDT, meilleurs counters, incremental repair, CQL devient mature)
- 20/07/2016: Cassandra 2.2 (essentiellement les UDF/UDA, support du JSON)
- 09/11/2015: Cassandra 3.0 (vues matérialisées, moteur de stockage ré-écrit, hints ré-implémentés,…)

Petit rappel historique des releases

- 03/09/2013: Cassandra 2.0 (Lightweight transactions)
- 16/09/2014: Cassandra 2.1 (UDT, meilleurs counters, incremental repair, CQL devient mature)
- 20/07/2016: Cassandra 2.2 (essentiellement les UDF/UDA, support du JSON)
- 09/11/2015: Cassandra 3.0 (vues matérialisées, moteur de stockage ré-écrit, hints ré-implémentés,…)
- Tick Tock release

Tick-Tock release

- Release mensuel
  - 1 mois pour introduire de nouvelles fonctionnalités
  - 1 mois pour corriger les bugs

- Objectifs
  - éviter l'effet tunnel des fonctionnalités qui s'accumulent pendant 6 mois/1 an sans feedback des utilisateurs
  - feedback plus rapide de la communauté

- Résultat
  - catastrophe
  - peu voir pas de feedback
  - des nouvelles fonctionnalités trop rapide donc instables

Tick-Tock release

- Qu'est ce qui a péché ?
  - les versions mises en production sont toujours la 2.1/2.2
  - sur la mailing-list officielle, il est conseillé d'utiliser la 2.1 …
  - personne ne se mouille pour tester les nouvelles fonctionnalités

Tick-Tock release

- Qu'est ce qui a péché ?
  - les versions mises en production sont toujours la 2.1/2.2
  - sur la mailing-list officielle, il est conseillé d'utiliser la 2.1 …
  - personne ne se mouille pour tester les nouvelles fonctionnalités

- Prémices des soucis à venir …
  - idéalisation excessive de la communauté
  - OSS = 100% des utilisateurs viennent se servir pour x% qui contribuent

Tick-Tock release

- Qu'est ce qui a péché ?
  - les versions mises en production sont toujours la 2.1/2.2
  - sur la mailing-list officielle, il est conseillé d'utiliser la 2.1 …
  - personne ne se mouille pour tester les nouvelles fonctionnalités

- Prémices des soucis à venir …
  - idéalisation excessive de la **communauté**
  - OSS = **100%** des utilisateurs viennent se servir pour **x%** qui contribuent

- Qu'est ce **contribuer** ?
  - publier de nouveau code
  - tester et donner du feedback
  - documentation, blog post, StackOverflow, évangélisation

Petit rappel historique des releases

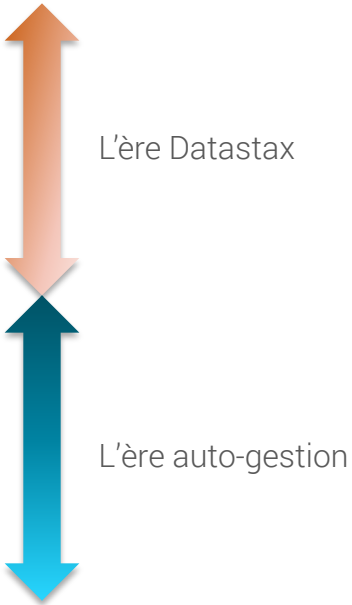
- 03/09/2013: Cassandra 2.0 (Lightweight transactions)
- 16/09/2014: Cassandra 2.1 (UDT, meilleurs counters, incremental repair, CQL devient mature)
- 20/07/2016: Cassandra 2.2 (essentiellement les UDF/UDA, support du JSON)
- 09/11/2015: Cassandra 3.0 (vues matérialisées, moteur de stockage ré-écrit, hints ré-implémentés,…)
- Tick Tock release
- 02/11/2016: Jonathan Ellis annonce un **désengagement relatif** de Datastax vis à vis d'Apache Cassandra™

Petit rappel historique des releases

- 23/06/2017: Cassandra 3.11 (stabilisations et nombreux bug fixs)
- ??/??/????: Cassandra 4.0

Petit rappel historique des releases

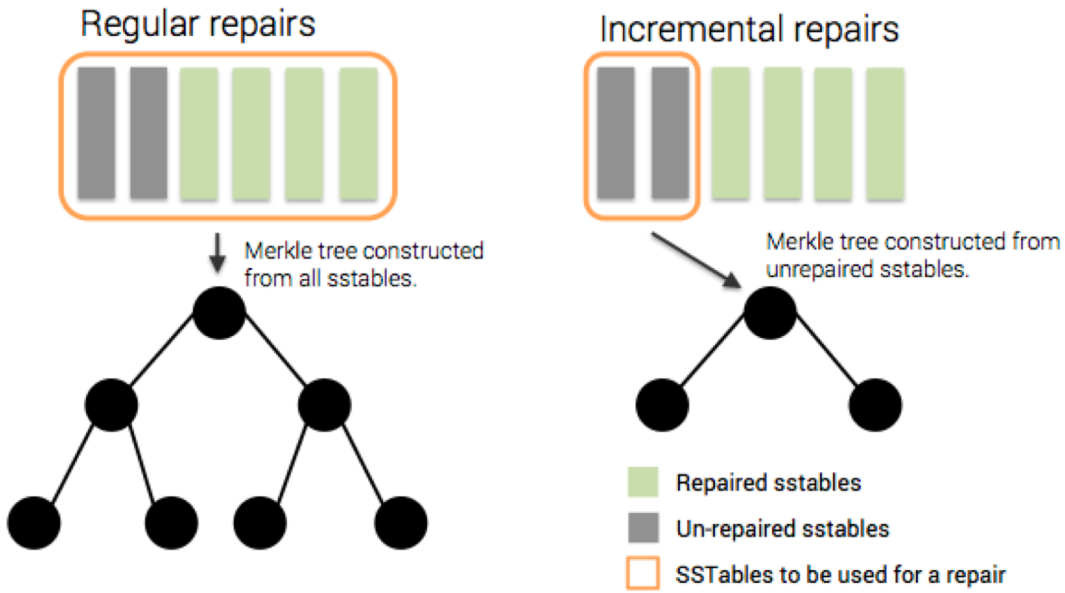
- 12/04/2010: Cassandra 0.6
- 02/11/2016: Désengagement relatif de Datastax (Cassandra 3.x)
- 23/06/2017: Cassandra 3.11
- 2018



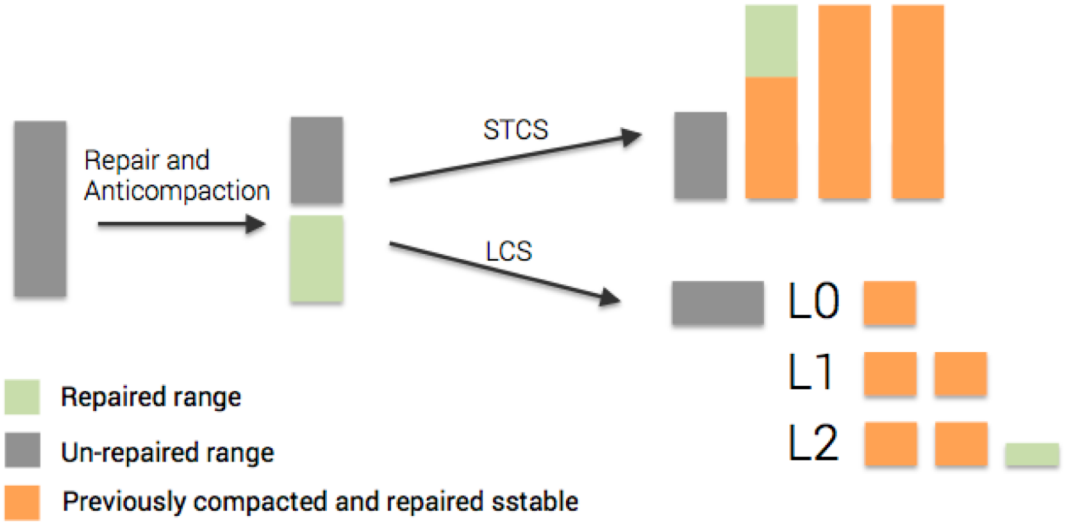
Roadmap 4.0 (committed)

- CASSANDRA-9425: Make node-local schema fully immutable
- CASSANDRA-12229 : Move streaming to non-blocking IO and netty
- CASSANDRA-9143: Fix consistency of incrementally repaired data across replicas

Incremental repair



Incremental repair



Roadmap 4.0 (committed)

- CASSANDRA-9425: Make node-local schema fully immutable
- CASSANDRA-12229 : Move streaming to non-blocking IO and netty
- CASSANDRA-9143: Fix consistency of incrementally repaired data across replicas
- CASSANDRA-14556: Faster streaming of SSTables using ZeroCopy APIs
- CASSANDRA-7396: Support for selecting Map values and Set elements

CASSANDRA-7396: Support for selecting Map values and Set elements

- Map dans Cassandra == SortedMap<K,V>

```
SELECT map[key] FROM table …

SELECT map[key1…key2] FROM table …
```

- Syntaxe pour les sets

```
SELECT set[key] FROM table …
```

- Et les lists ???

Roadmap 4.0 (committed)

- CASSANDRA-9425: Make node-local schema fully immutable
- CASSANDRA-12229 : Move streaming to non-blocking IO and netty
- CASSANDRA-9143: Fix consistency of incrementally repaired data across replicas
- CASSANDRA-14556: Faster streaming of SSTables using ZeroCopy APIs
- CASSANDRA-7396: Support for selecting Map values and Set elements
- CASSANDRA-7461: Add support for arithmetic operators
- CASSANDRA-11936: Add support for + and – operations on dates (CASSANDRA-11873 add duration type)

CASSANDRA-11936: Add support for + and – operations on dates

- Types d'opérations supportés:
  - date + duration
  - date - duration
  - timestamp + duration
  - timestamp - duration

- Utilisation

```
SELECT … FROM table WHERE date > now() - 2h
```

Roadmap 4.0 (committed)

- CASSANDRA-9425: Make node-local schema fully immutable
- CASSANDRA-12229 : Move streaming to non-blocking IO and netty
- CASSANDRA-9143: Fix consistency of incrementally repaired data across replicas
- CASSANDRA-14556: Faster streaming of SSTables using ZeroCopy APIs
- CASSANDRA-7396: Support for selecting Map values and Set elements
- CASSANDRA-7461: Add support for arithmetic operators
- CASSANDRA-11936: Add support for + and – operations on dates (CASSANDRA-11873 add duration type)
- CASSANDRA-7622: Virtual tables

CASSANDRA-7622: Virtual tables

- Tables prédéfinies dans keyspace system_info:
  - table_stats
  - ring_state
  - compaction_stats
  - settings
- Utilisation

```
SELECT * FROM system_info.settings WHERE writable = True;

setting                          | value | writable
---------------------------------+-------+----------
batch_size_fail_threshold_in_kb | 50    | True
batch_size_warn_threshold_in_kb | 5     | True
cas_contention_timeout_in_ms    | 1000  | True
…
```

CASSANDRA-7622: Virtual tables

- Utilisation (suite)

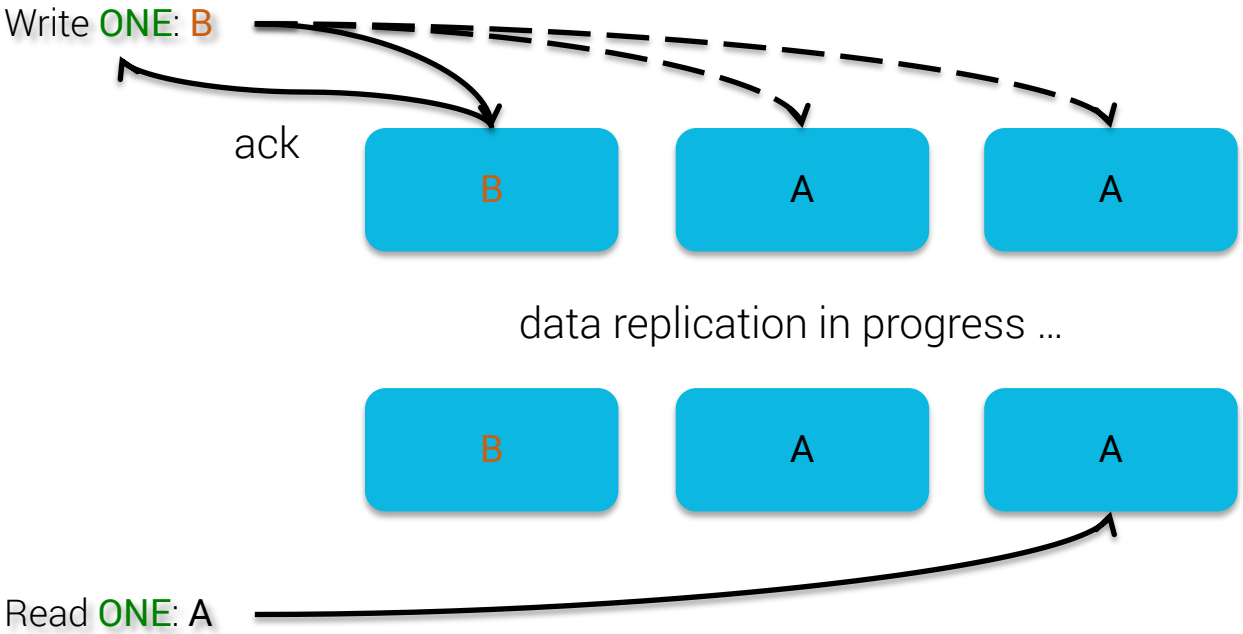
```
SELECT keyspace_name, table_name, metric, median, p99th FROM system_info.table_stats
WHERE keyspace_name = 'system' and p99th > 0;

keyspace_name | table_name         | metric                           | median     | p99th
--------------+--------------------+----------------------------------+------------+------------
system        | batches            | rangeLatency                     | 1.5232e+05 | 3.7902e+05
system        | compaction_history | estimatedColumnCountHistogram    | 6          | 7
system        | compaction_history | estimatedPartitionSizeHistogram  | 124        | 149
system        | local              | estimatedColumnCountHistogram    | 17         | 310
system        | local              | estimatedPartitionSizeHistogram  | 215        | 6866
```

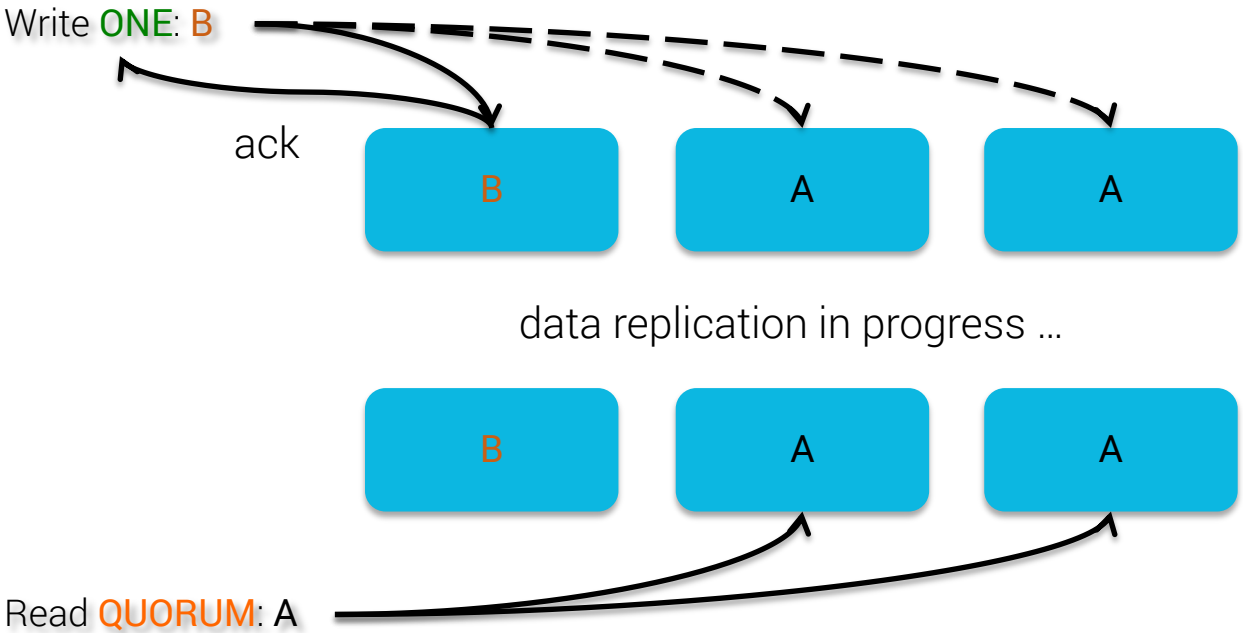
Roadmap 4.0 (committed)

- CASSANDRA-9425: Make node-local schema fully immutable
- CASSANDRA-12229 : Move streaming to non-blocking IO and netty
- CASSANDRA-9143: Fix consistency of incrementally repaired data across replicas
- CASSANDRA-14556: Faster streaming of SSTables using ZeroCopy APIs
- CASSANDRA-7396: Support for selecting Map values and Set elements
- CASSANDRA-7461: Add support for arithmetic operators
- CASSANDRA-11936: Add support for + and – operations on dates (CASSANDRA-11873 add duration type)
- CASSANDRA-7622: Virtual tables
- CASSANDRA-13289: Ideal consistency level
- CASSANDRA-14404: Transient replica & Cheap Quorum

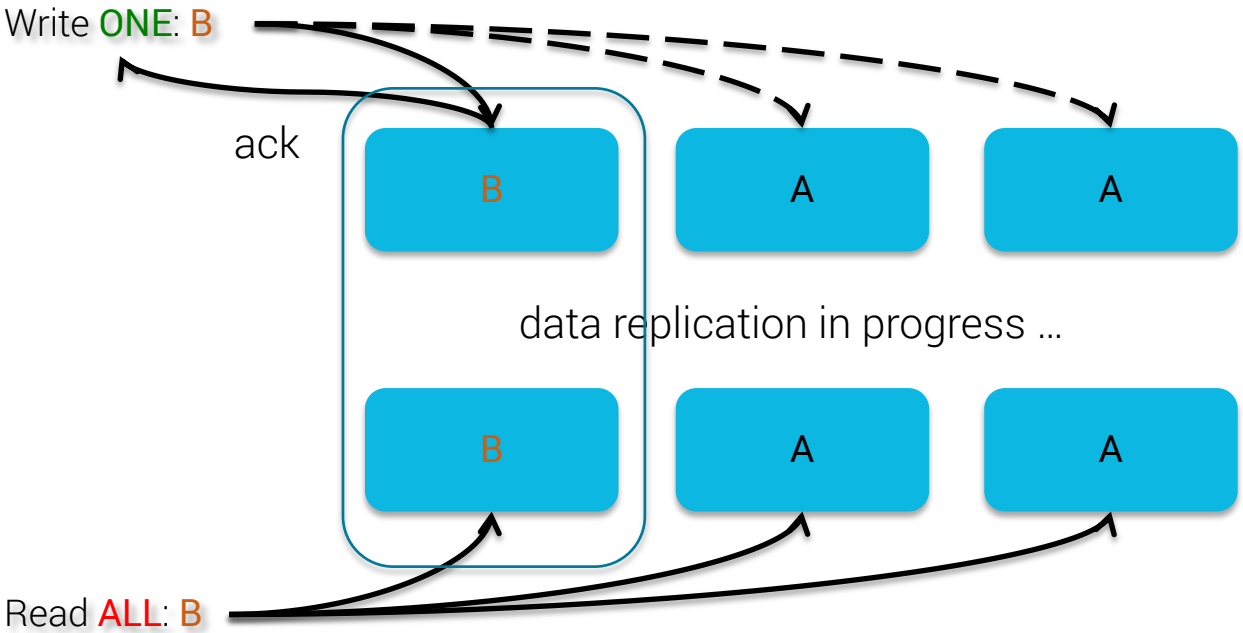
Rappel sur les consistency level



Rappel sur les consistency level

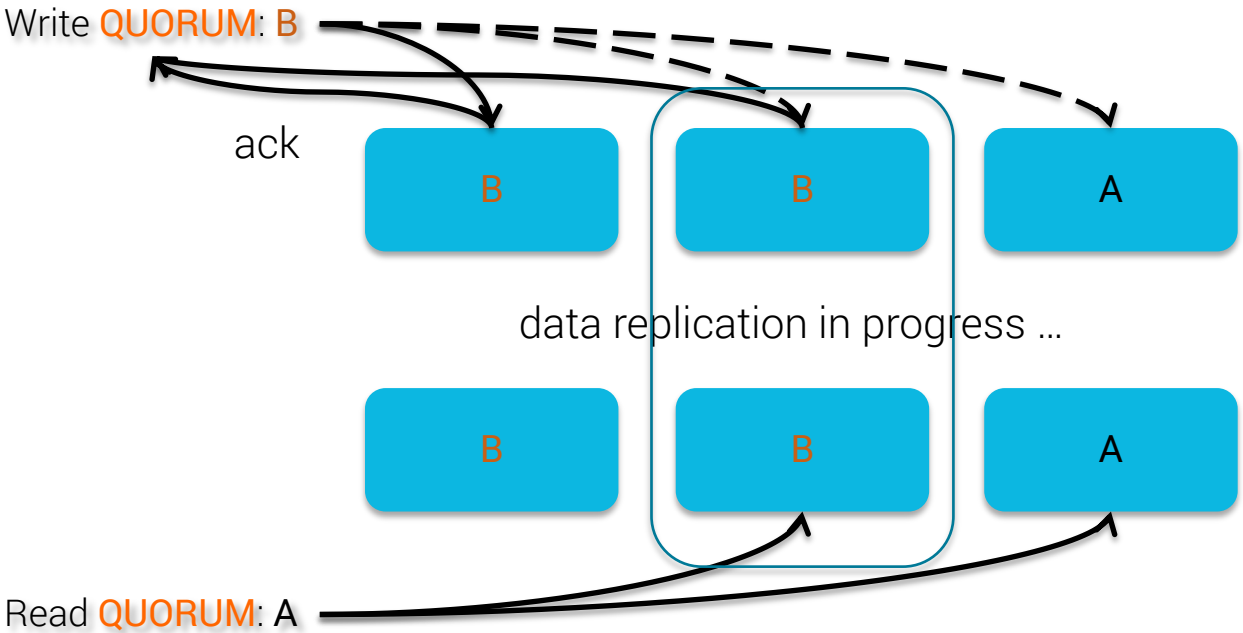


Rappel sur les consistency level

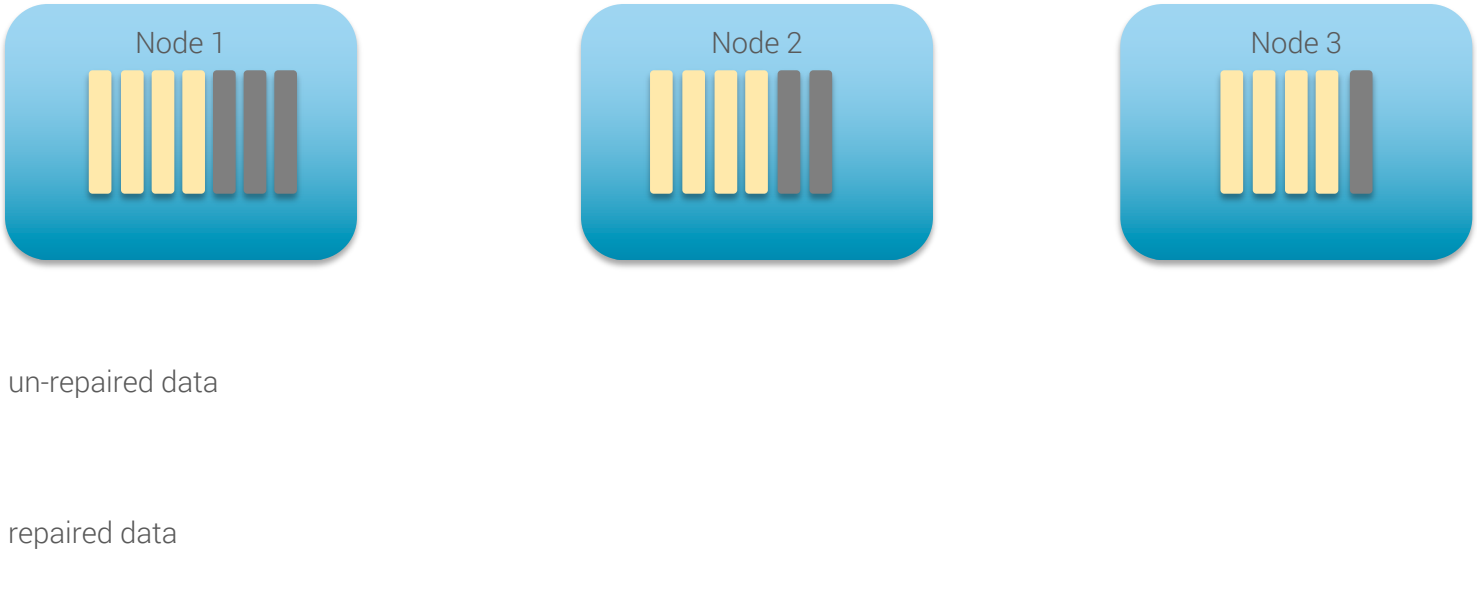


Rappel sur les consistency level

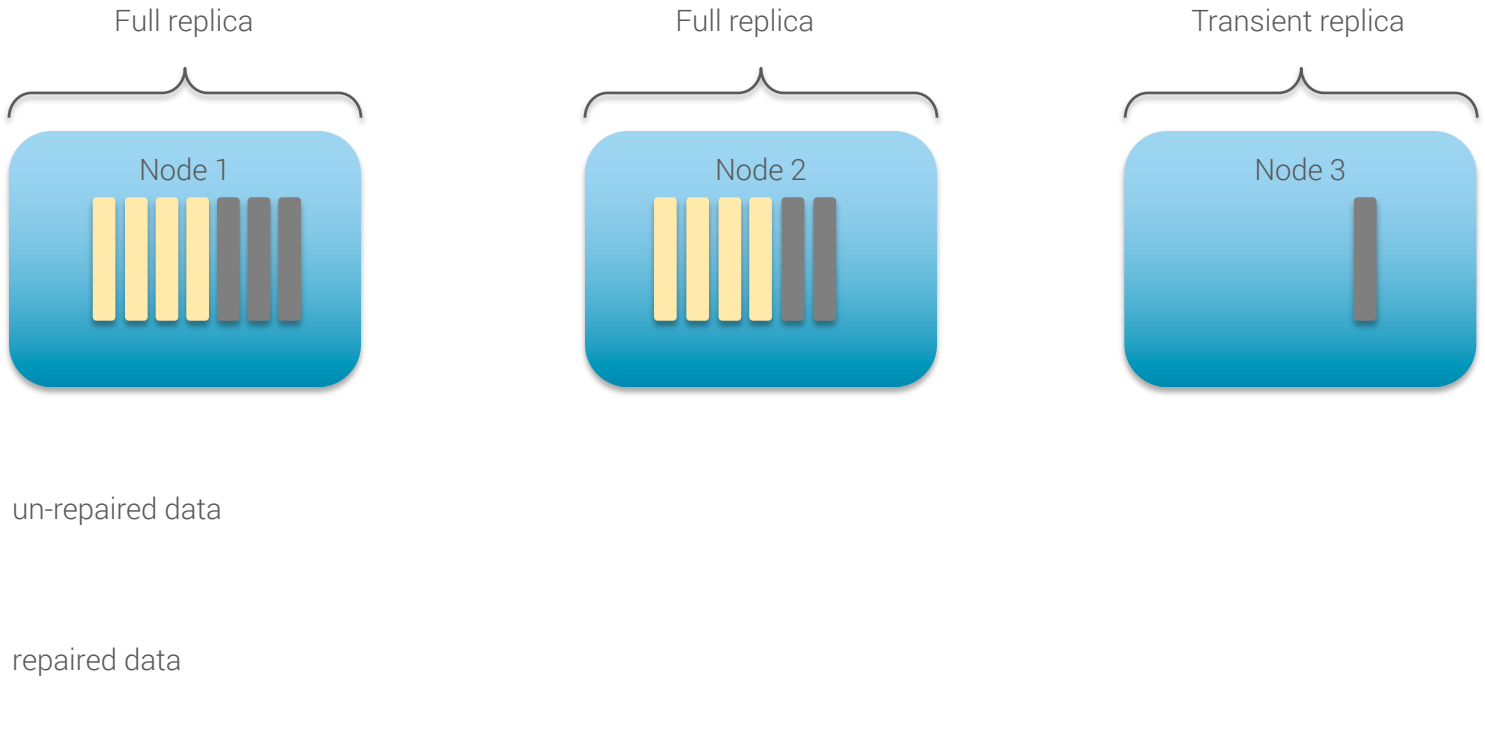
RF = 3, Write QUORUM, Read QUORUM



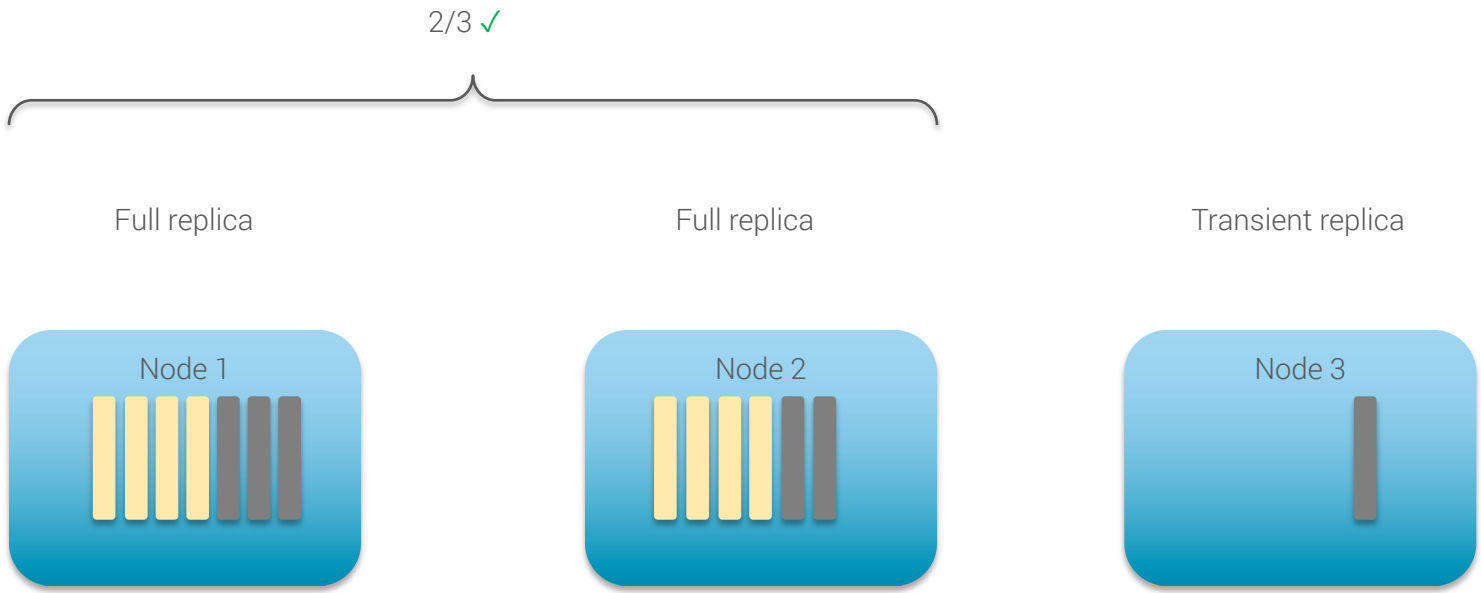
Transient replica: idées



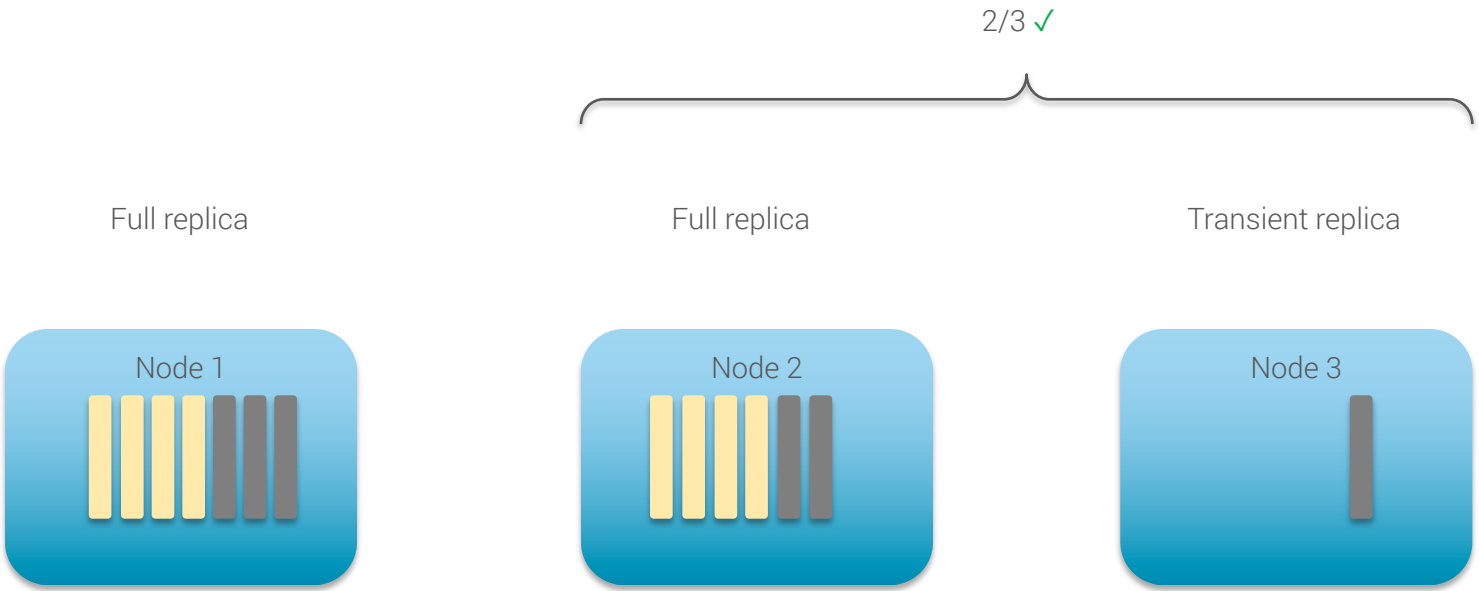
Transient replica: idées



Transient replica: scénario lecture/écriture QUORUM/LOCAL_QUORUM

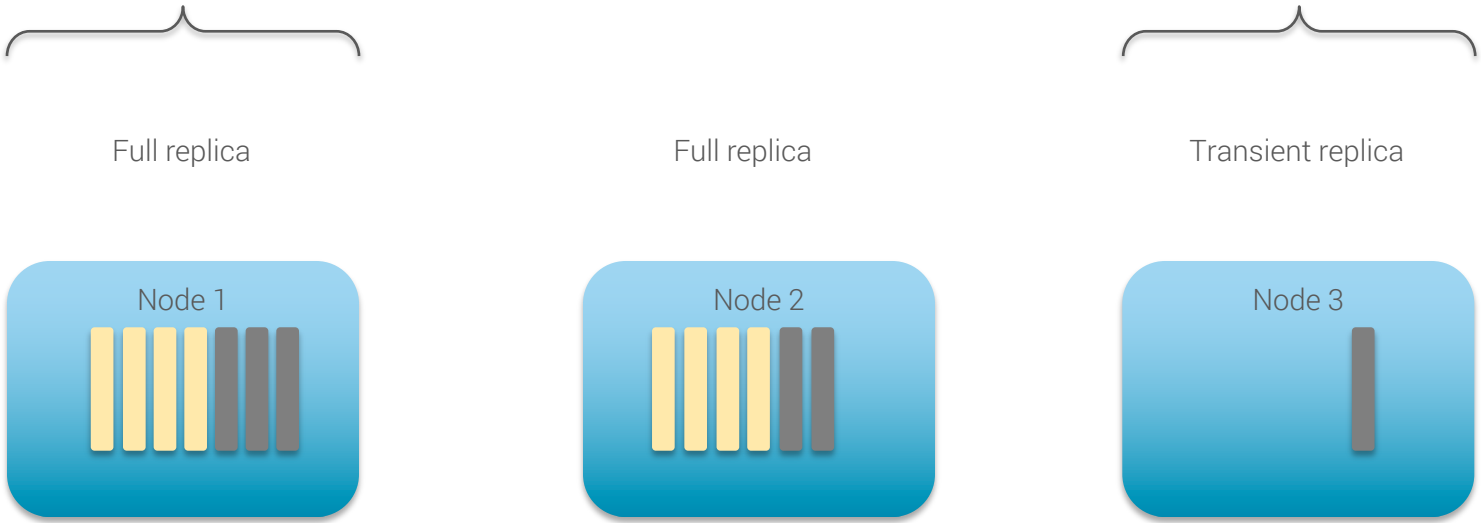


Transient replica: scénario lecture/écriture QUORUM/LOCAL_QUORUM

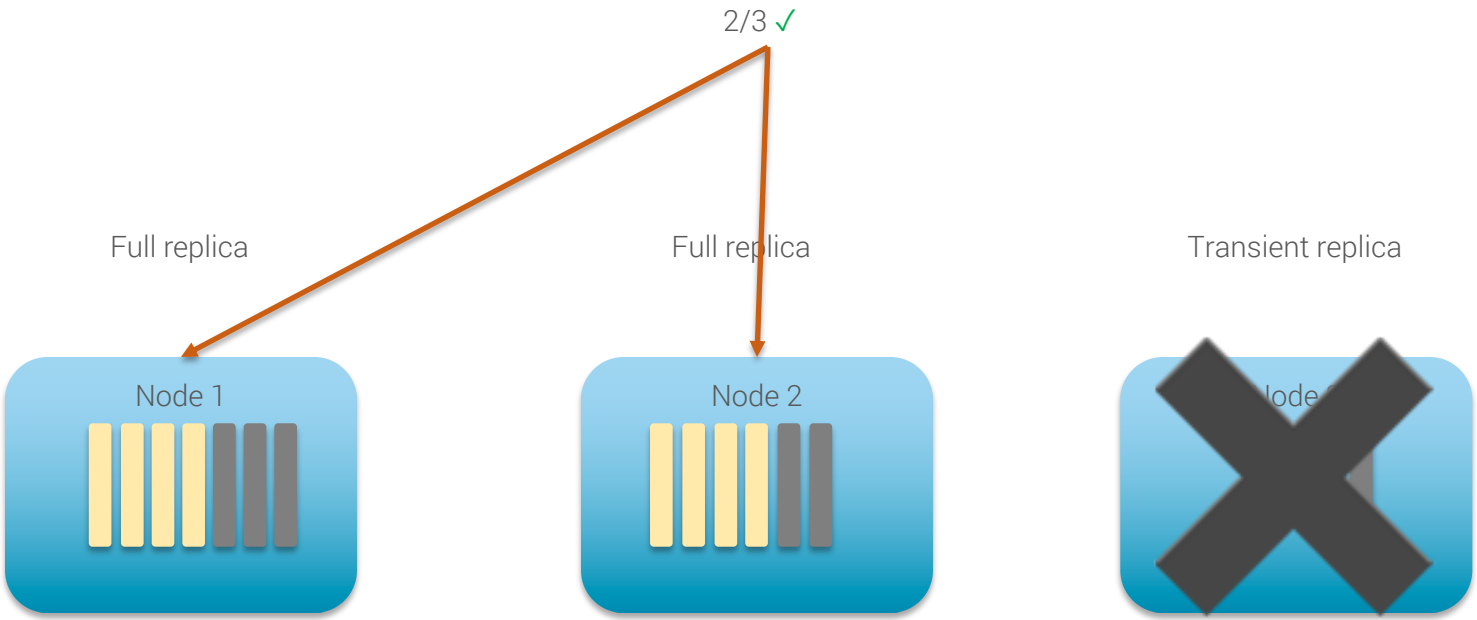


Transient replica: scénario lecture/écriture QUORUM/LOCAL_QUORUM

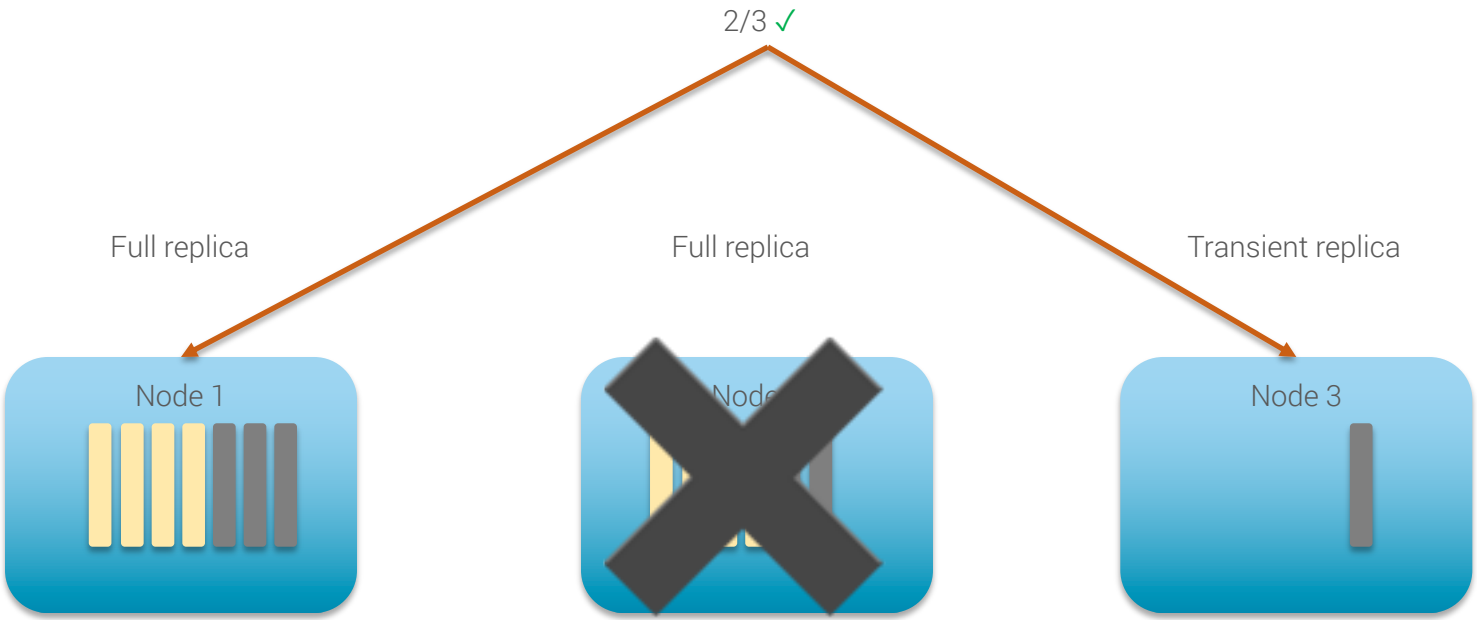
2/3 ✓



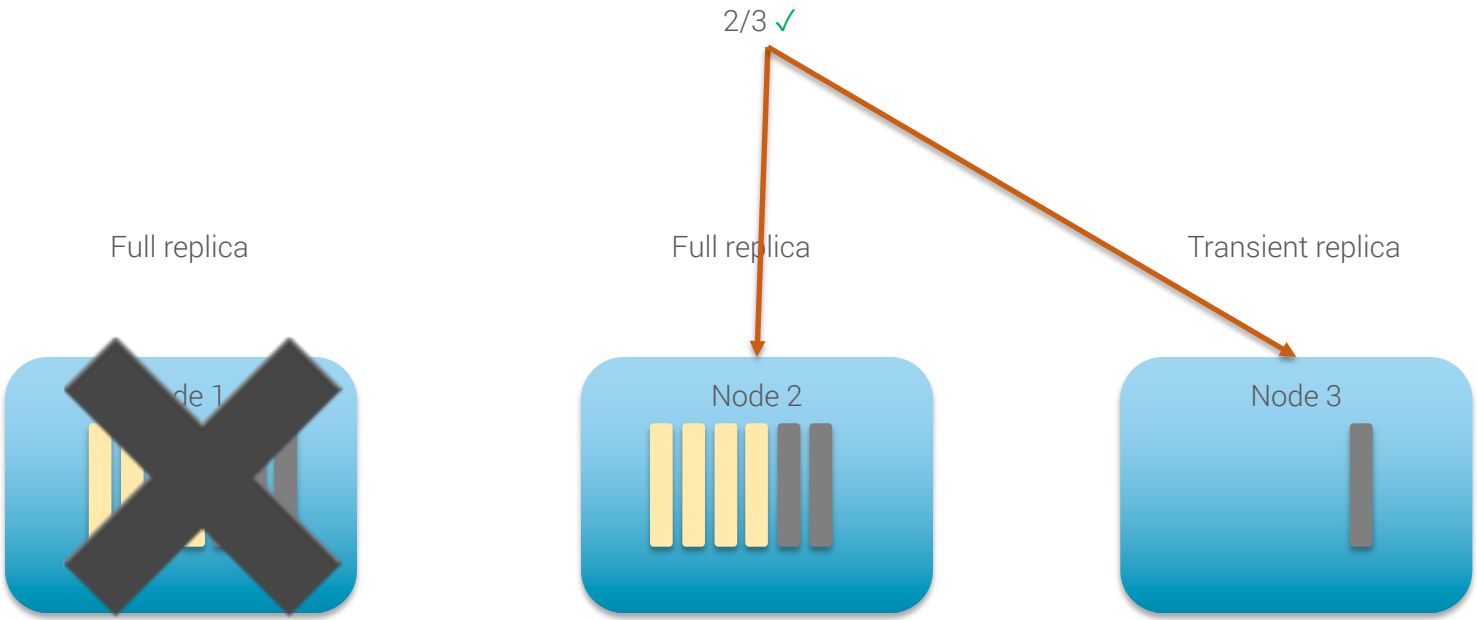
Transient replica: scénario haute dispo QUORUM/LOCAL_QUORUM



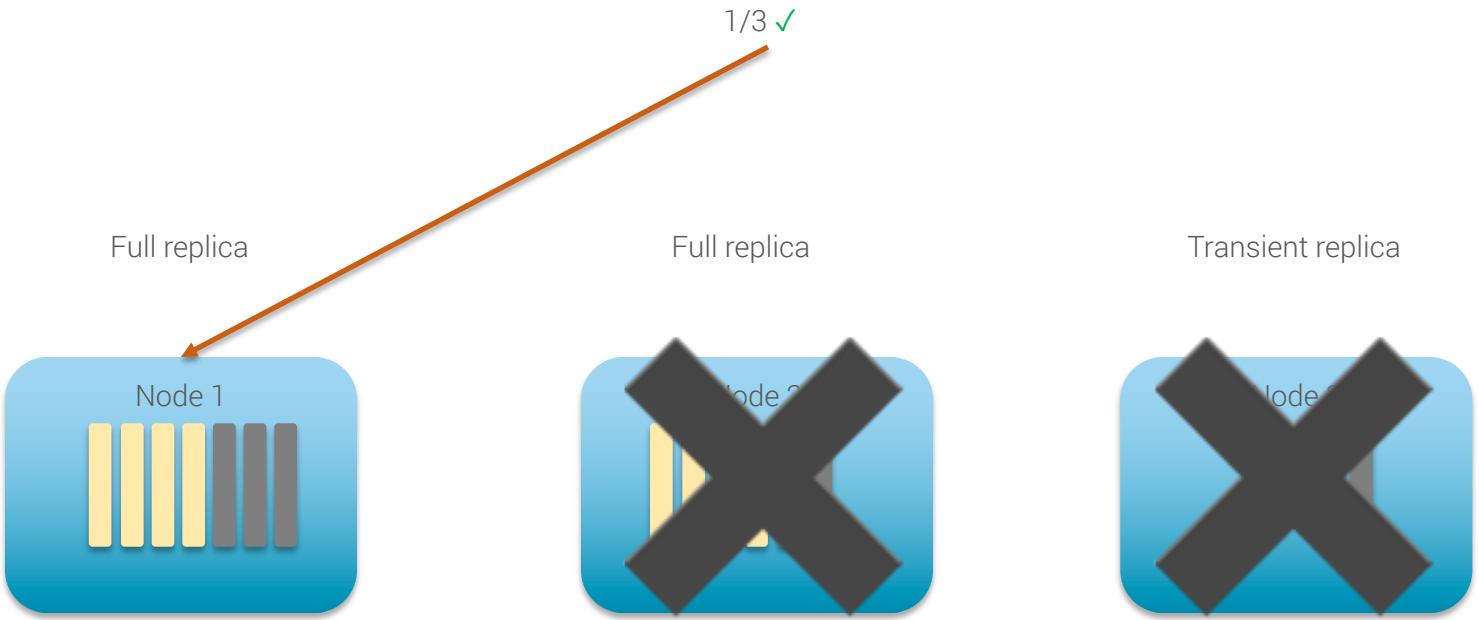
Transient replica: scénario haute dispo QUORUM/LOCAL_QUORUM



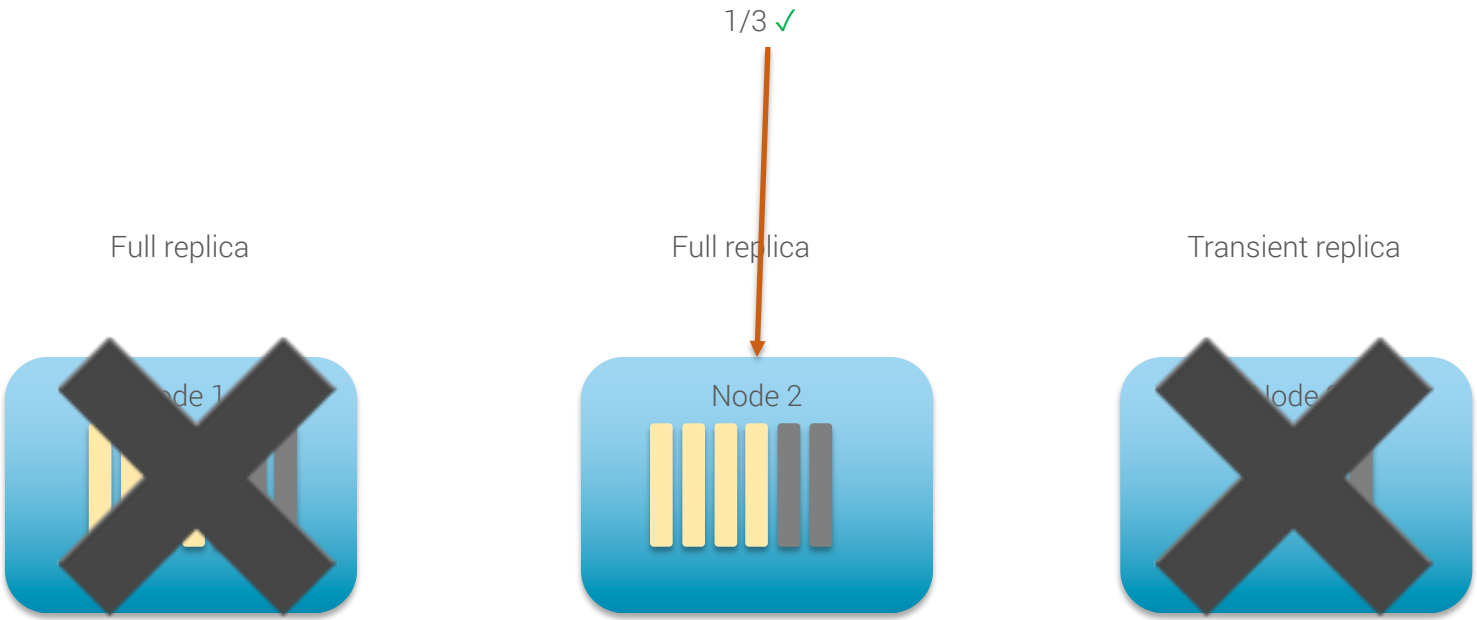
Transient replica: scénario haute dispo QUORUM/LOCAL_QUORUM



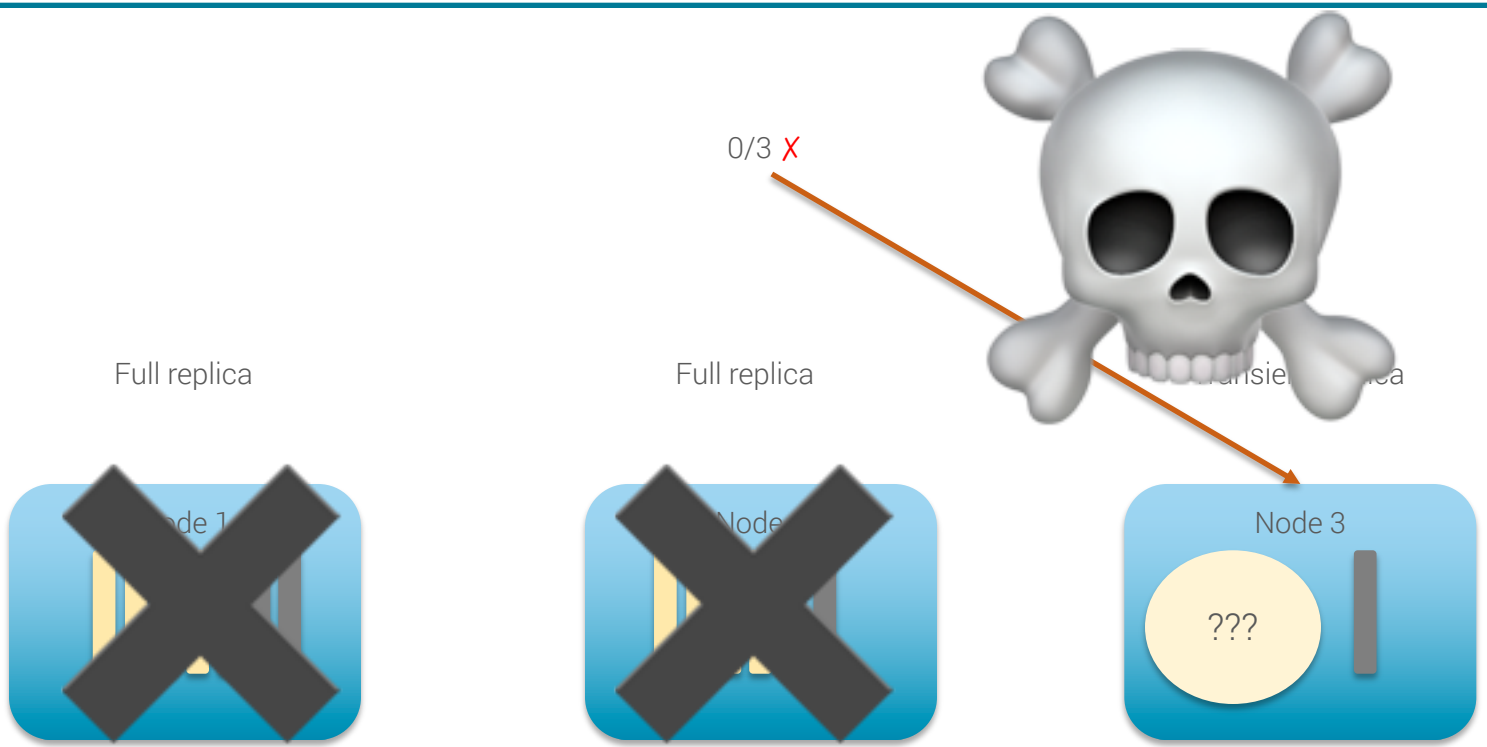
Transient replica: scénario haute dispo ONE/LOCAL_ONE



Transient replica: scénario haute dispo ONE/LOCAL_ONE



Transient replica: scénario haute dispo ONE/LOCAL_ONE



Transient replica: scénario haute dispo ONE/LOCAL_ONE

Voting with Witnesses: A Consistency Scheme for Replicated Files

Jehan-François Pâris[†]

Computer Systems Research Group
Department of Electrical Engineering and Computer Sciences
University of California, San Diego
La Jolla, California 92093

International Conference on Distributed Computing Systems, 1986, pages 606–612

copies and one witness being constantly updated. As one can see, the availability of a replicated file with two copies and one witness remains very close to the availability of a replicated file with three copies as long as ρ remains small. This will be the

where $\rho = \frac{\lambda}{\mu}$ is the ratio of the failure rate over the repair rate.

Roadmap 4.0 (committed)

- CASSANDRA-14404: Transient replica & Cheap Quorum
  - trop spécifique au gros clusters/multi-DC (spécifique Apples)

Roadmap 4.0 (committed)

- CASSANDRA-14404: Transient replica & Cheap Quorum
  - trop spécifique au gros clusters/multi-DC (spécifique Apples)
  - incompatible par nature avec vues matérialisées et 2nd index

Roadmap 4.0 (committed)

- CASSANDRA-14404: Transient replica & Cheap Quorum
  - trop spécifique au gros clusters/multi-DC (spécifique Apples)
  - incompatible par nature avec vues matérialisées et 2nd index
  - incremental repair obligatoire … (pour augmenter le taux repair rate vs failure rate)

Roadmap 4.0 (committed)

- CASSANDRA-14404: Transient replica & Cheap Quorum
  - trop spécifique au gros clusters/multi-DC (spécifique Apples)
  - incompatible par nature avec vues matérialisées et 2nd index
  - incremental repair obligatoire … (pour augmenter le taux repair rate vs failure rate)
  - très dûr à raisonner sans un doctorat en informatique distribué

Roadmap 4.0 (committed)

- CASSANDRA-14404: Transient replica & Cheap Quorum
  - trop spécifique au gros clusters/multi-DC (spécifique Apples)
  - incompatible par nature avec vues matérialisées et 2nd index
  - incremental repair obligatoire … (pour augmenter le taux repair rate vs failure rate)
  - très dûr à raisonner sans un doctorat en informatique distribué
  - trous dans la raquette pour le cas ONE/LOCAL_ONE

Roadmap 4.0 (committed)

- CASSANDRA-14404: Transient replica & Cheap Quorum
  - trop spécifique au gros clusters/multi-DC (spécifique Apples)
  - incompatible par nature avec vues matérialisées et 2nd index
  - incremental repair obligatoire … (pour augmenter le taux repair rate vs failure rate)
  - très dûr à raisonner sans un doctorat en informatique distribué
  - trous dans la raquette pour le cas ONE/LOCAL_ONE
  - introduit une dissymétrie parmi les nœuds

Roadmap 4.0 (committed)

- CASSANDRA-14404: Transient replica & Cheap Quorum
  - trop spécifique au gros clusters/multi-DC (spécifique Apples)
  - incompatible par nature avec vues matérialisées et 2nd index
  - incremental repair obligatoire … (pour augmenter le taux repair rate vs failure rate)
  - très dûr à raisonner sans un doctorat en informatique distribué
  - trous dans la raquette pour le cas ONE/LOCAL_ONE
  - introduit une dissymétrie parmi les nœuds
  - impact énorme sur la base de code

Roadmap 4.0 (committed)

- CASSANDRA-14404: Transient replica & Cheap Quorum
  - état d'avancement

Sub-Tasks

1.	Transient Replication: Metadata refactor		RESOLVED	Blake Eggleston
2.	Transient Replication: Implement cheap quorum write optimizations		RESOLVED	Blake Eggleston
3.	Transient Replication: Add support for correct reads when transient replication is in use		RESOLVED	Blake Eggleston
4.	Transient Replication: Incremental & Validation repair handling of transient replicas		RESOLVED	Blake Eggleston
5.	Transient Replication: Support ring changes when transient replication is in use (add/remove node, change RF, add/remove DC)		RESOLVED	Ariel Weisberg
6.	Transient Replication: Support replication factor changes		OPEN	*Unassigned*
7.	Transient Replication: Support paxos		OPEN	*Unassigned*
8.	Transient Replication: support counters		OPEN	*Unassigned*
9.	Transient Replication: support logged batches		OPEN	*Unassigned*
10.	Transient Replication: Support monotonic reads		OPEN	*Unassigned*
11.	Transient Replication: Confirm vnode support w/Transient Replication		OPEN	*Unassigned*

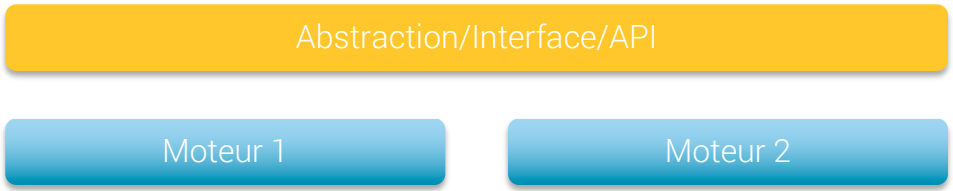
Roadmap 4.0 (not-committed)

- CASSANDRA-9754: Making index friendlier for large partitions (awaiting feedback 17/02/2017)
- CASSANDRA-12345 : Gossip 2.0 (spécifique Apple, 31/08/2016)
- CASSANDRA-10540: Range-aware compaction (dernier échange 04/09/2018)
- CASSANDRA-13474: Pluggable storage engine (spécifique Instagram)

CASSANDRA-13474: Pluggable storage engine

- Quoi ?
  - proposer différentes implémentations pour le moteur de stockage bas niveau
  - actuellement, le moteur date du refactoring 3.0 et est adapté à l'abstraction Map<K, SortedMap<K,V>>
  - but == intégrer RocksDB comme moteur de stockage alternatif

- Comment faire ?
  - commencer à refactorer tout le code pour isoler les niveaux d'abstraction



CASSANDRA-13474: Pluggable storage engine

- Etat d'avancement

Sub-Tasks

1.	Pluggable storage engine design		OPEN	Dikang Gu
2.	Refactor streaming		RESOLVED	Blake Eggleston
3.	Refactor repair		RESOLVED	Blake Eggleston
4.	Refactor read path		PATCH AVAIL...	Dikang Gu
5.	Refactor write path		RESOLVED	Blake Eggleston
6.	Refactor compaction		OPEN	*Unassigned*
7.	Refactor Keyspace/CFS operations		OPEN	*Unassigned*
8.	Refactor metrics		OPEN	*Unassigned*
9.	Refactor Indexes		OPEN	*Unassigned*
10.	Abstract storage engine API from Keyspace/CFS		OPEN	*Unassigned*
11.	Refactor Schema/Metadata		OPEN	*Unassigned*
12.	Refactor commitlog		OPEN	*Unassigned*

CASSANDRA-13474: Pluggable storage engine

- Pour
  - moteurs adaptés à des besoins différents
  - force à refactorer le code proprement

- Contre
  - problème à l'origine spécifique à **Instagram**
  - chantier **titanesque** (GA = 2020 au mieux)
  - **risque élevé** inhérent à tout refacto (cf refacto 3.0 et régression de performance)
  - quid du **support des fonctionnalités** spécifiques au modèle Map<K,SortedMap<K,V>> ?
    - UDT
    - collections (map, set, lists)
    - counters
    - static columns
    - …

CASSANDRA-13474: Pluggable storage engine

- Contre (suite)
  - risque d'aboutir à une matrice de fonctionnalités gruyère

	Fonc.1	Fonc.2	Fonc.3	Fonc.4	Fonc.5	Fonc.6	Fonc.7
Moteur1	✓	✓	✓	✓	✓	✗	✗
Moteur2	✗	✗	✓	✓	✓	✓	✗
Moteur3	✓	✓	✓	✗	✗	✓	✓
Moteur4	✗	✗	✗	✓	✓	✓	✓

  - devoir gérer RockDB (compaction, ttl …)
  - devoir gérer différent languages (C++ pour RocksDB par ex)

Roadmap 4.0 (not-committed)

- CASSANDRA-9754: Making index friendlier for large partitions (awaiting feedback 17/02/2017)
- CASSANDRA-12345 : Gossip 2.0 (spécifique Apple, 31/08/2016)
- CASSANDRA-10540: Range-aware compaction (dernier échange 04/09/2018)
- CASSANDRA-13474: Pluggable storage engine (spécifique Instagram)
- CASSANDRA-13001: Pluggable slow query log/handling (dernier échange 21/03/2018)
- CASSANDRA-12106: Ability to blacklist a bad partition (dernier échange 07/09/2018)
- CASSANDRA-14395: Cassandra management process (clash Netflix vs TLP)

Roadmap 4.0 conclusion

- Quelques fonctionnalités intéressantes

Roadmap 4.0 conclusion

- Quelques fonctionnalités intéressantes
- Beaucoup de tickets encore ouverts/en attente de revue

Roadmap 4.0 conclusion

- Quelques fonctionnalités intéressantes
- Beaucoup de tickets encore ouverts/en attente de revue
- Beaucoup de demande de feedback à la communauté, peu de retour/d'engagements

Roadmap 4.0 conclusion

- Quelques fonctionnalités intéressantes
- Beaucoup de tickets encore ouverts/en attente de revue
- Beaucoup de demande de feedback à la communauté, peu de retour/d'engagements
- Il est difficile de contribuer du code, encore plus pour faire du code-review sur les nouvelles contributions et surtout la QA

La fragmentation de l’éco-système

Les acteurs en présence

- OSS Cassandra
- Datastax Entreprise (divergera de plus en plus)
- ScyllaDB (le clone malhonnête)
- Fork interne chez Apple (tends à disparaître si Apple prend le contrôle du projet OSS)
- Fork interne chez Instagram (disparaitra si Pluggable Storage Engine fini complètement)
- Elassandra (petit produit de chez nous Cocorico)
- YugaByte
- CosmoDB

OSS Cassandra

- On ne le présente plus
- Continuera à vivre quoiqu'il arrive
- Doutes sur:
  - rythme des nouvelles innovations
  - pertinence des nouvelles fonctionnalités (centrées sur Apple/poweruser seulement ?)
  - équilibre dev vs ops

Datastax Entreprise

- Core OSS Cassandra
- + features propriétaires (NodeSync (continuous repair), InMemory, BackPressure, Thread Per Core)
- + intégrations propriétaires:
  - Apache Spark (haute dispo et bientôt masterless)
  - Apache Solr (haute dispo et déjà masterless)
  - Graphe avec Apache Tinkerpop
- Divergera de + en +
- Datastax reprendra certainement des nouvelles fonctionnalités contribuées par d'autre à OSS Cassandra

ScyllaDB

- Copier-coller de Cassandra en C++
- Le plus détestable car marketing agressif et mensonger (drop-in replacement de Cassandra, faux)
- Marché de niche, seule différence = performance mais le Thread Per Core de Datastax change la donne
- Communauté quasi inexistante
- Risques de bugs/régression à cause de la ré-écriture en C++ (question de QA)

Forks interns chez Apple et Instagram

- Fonctionnalités répondant aux besoins spécifiques
- Vont converger avec OSS Cassandra, ou pas …

Elassandra (made in France)

- Mariage d'Elastic Search et OSS Cassandra
- Données brutes dans Cassandra, index dans ES
- Tabasse Elastic Search en terme de scalabilité
  - on peut enfin stocker ses données dans une vrai base de données
  - on peut resharder comme on veut!
  - architecture masterless de Cassandra !!!
- Donne un gros boost à Cassandra en terme d'outillage de recherche
  - toute l'API d'ES
  - tout l'écosystème d'ES (ELK) disponible
- Mais …
  - projet d'un seul homme (Vincent Royer) …
  - … qui n'est ni committer Cassandra ni Elastic Search
  - … exploite quelques détails d'implémentation de Cassandra
  - donc à la merci de tout changement d'architecture de ces 2 produits

YugaByte

- RocksDB fortement modifié comme storage engine
- supporte l'API CQL de Cassandra
- couche distribuée utilisant Raft
- parasitisme d'API

CosmoDB

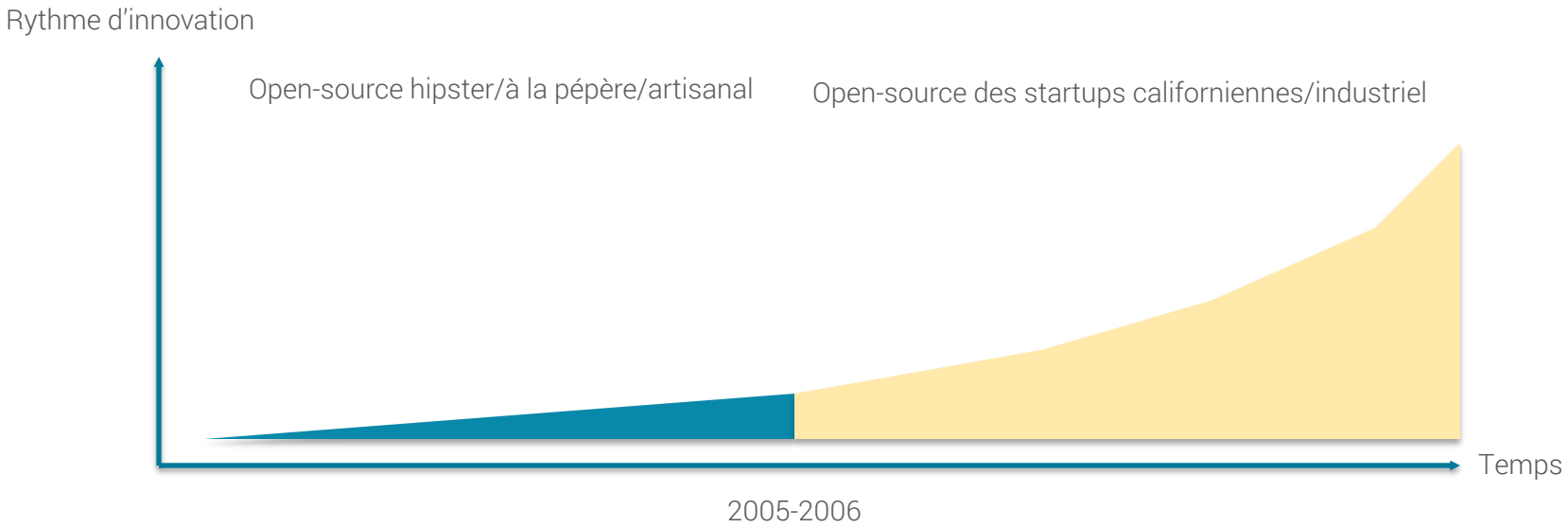
- parasitisme d'API par Grosoft
- supporte l'API CQL de Cassandra …
- … ainsi que celui de Mongo

NOSQLWARS

EPISODE I

LA MENACE DU 'CLOUD'

Historique de l’open-source



La période des startups

- but des startups technologique == gagner de l'argent (pardon pour le truisme)
- comment imposer une nouvelle technologie ?
- comment accélérer l'adoption ?

La période des startups

- but des startups technologique == gagner de l'argent (pardon pour le truisme)
- comment imposer une nouvelle technologie ?
- comment accélérer l'adoption ?
- ☞ Open Source !!!
  - bénéficie d'une bonne image auprès des développeurs naïfs

La période des startups

- but des startups technologique == gagner de l'argent (pardon pour le truisme)
- comment imposer une nouvelle technologie ?
- comment accélérer l'adoption ?
- ☞ Open Source !!!
  - bénéficie d'une bonne image auprès des développeurs naïfs
  - ouvert donc transparent/auditable donc peu de risque de se lier

La période des startups

- but des startups technologique == gagner de l'argent (pardon pour le truisme)
- comment imposer une nouvelle technologie ?
- comment accélérer l'adoption ?
- ☞ Open Source !!!
  - bénéficie d'une bonne image auprès des développeurs naïfs
  - ouvert donc transparent/auditable donc peu de risque de se lier
  - gratuit !!! 🤑 🤑 🤑

La période des startups

- but des startups technologique == gagner de l'argent (pardon pour le truisme)
- comment imposer une nouvelle technologie ?
- comment accélérer l'adoption ?
- ☞ Open Source !!!
  - bénéficie d'une bonne image auprès des développeurs naïfs
  - ouvert donc transparent/auditable donc peu de risque de se lier
  - gratuit !!! 🤑 🤑 🤑
  - contributions externes (de la communauté), enfin soi-disant …

La période des startups

- but des startups technologique == gagner de l'argent (pardon pour le truisme)
- comment imposer une nouvelle technologie ?
- comment accélérer l'adoption ?
- ☞ Open Source !!!
  - bénéficie d'une bonne image auprès des développeurs naïfs
  - ouvert donc transparent/auditable donc peu de risque de se lier
  - gratuit !!! 🤑 🤑 🤑
  - contributions externes (de la communauté), enfin soi-disant …
  - QA externalisée (si le projet est populaire)

Business models de l’open source

- but == comment transformer le gratuit en 💵 💵 💵 ?
- plusieurs possibilités:
  - support
  - consulting
  - outillage pour la prod et le monitoring
  - fonctionnalités supplémentaires
  - SaaS
  - fondations philanthropiques/financement participatif

Business models du support

- non pérenne sur le long terme
  - pourquoi payer quand on a des experts maisons ?
- incitation à rajouter de la complexité au produit
  - pour justifier l'intérêt du Support
- concurrence à terme par des anciens devenus experts
  - 10 anciens de Datastax montent une boite de consulting avec prix/2
  - exemple concret == The Last Pickle
- croissance du CA très linéaire et faible
  - comment justifier un coût de support démentiel vs le coût d'une vraie licence logicielle ?

Business models du consulting

- trop de concurrence
  - gros acteurs existants sur le marché (Accenture, Cap Gemini) qui imposent une grille de prix

- temps dédié à la R&D minuscule
  - maximisation du taux horaire consulting == minimisation du temps restant pour la R&D

- business très difficile à mettre à l'échelle
  - 100 clients = 20 consultants
  - 1000 clients ~ 200 consultants
  - perte d'un gros client == licencier xxx consultants ?

Business models de l'outillage

- marche plus ou moins bien
  - le cas de la plupart des acteurs du monde NoSQL/Big Data

- marche jusqu'à un certain point
  - empiète sur le marché des professionnels du monitoring (AppDynamics, DynaTrace …)

- risque de compétition par des développements open-source
  - coût de devs de l'outillage devient plus abordable
  - ex: plugin sécurité OSS pour ElasticSearch
  - ex: Cassandra Reaper de TheLastPickle qui concurrence partiellement OpsCenter de Datastax

- croissance du CA très linéaire et faible
  - comment justifier un coût premium vs le coût d'une vraie licence logicielle ?

Business models des fonctionnalités

- marche plutôt bien
  - jusqu'à un certain point …
- crée de la frustration chez les ayatollahs de l'open-source
- arbitrage délicat entre OSS et propriétaire
- risque de conflit d'intérêt dans le développement du core OSS
  - introduire une nouvelle API dans OSS
  - implémentation réelle dans la version entreprise
- modèle de Datastax jusqu'à récemment

Business models autour du SaaS

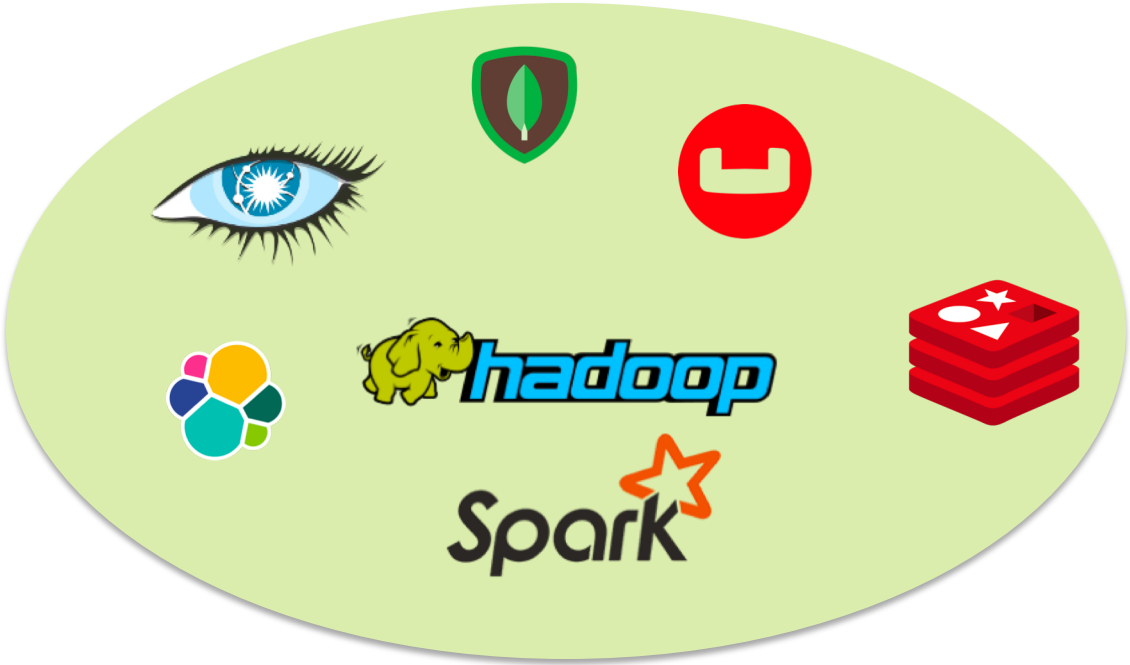
- la vraie réponse !!!
  - *« voulez vous payer une licence ? »* Jamais je ne paierai !
  - *« voulez vous payer un service ? »* Avec plaisir, voici ma CB
  - merci au travail cognitif d'Amazon sur la masse des développeurs

Business models autour du SaaS

- la vraie réponse !!!
  - *« voulez vous payer une licence ? »* Jamais je ne paierai !
  - *« voulez vous payer un service ? »* Avec plaisir, voici ma CB
  - merci au travail cognitif d'Amazon sur la masse des développeurs

- … mais il est trop tard !!!

Business models autour du SaaS



L'étang du NoSQL/Big Data

Business models autour du SaaS

L'étang du NoSQL/Big Data

Le lac du Cloud

La menace du « Cloud »

- le problème du parasitage des acteurs du cloud
  - *« Merci les gars d'avoir fait la R&D pour nous, on va packager tout ça pour en faire un service payant !!!»*

La menace du « Cloud »

- le problème du parasitage des acteurs du cloud
  - *« Merci les gars d'avoir fait la R&D pour nous, on va packager tout ça pour en faire un service payant !!!»*
  - ex ?
    - ElasticSearch sur AWS (merci Elastic)
    - Redis sur AWS (merci Redis Lab)
    - Hadoop sur AWS, Azure & GCP (merci Hortonworks, Cloudera & MapR)
    - Glue sur AWS, Spark sur GCP (merci DataBricks)

La menace du « Cloud »

- le problème du parasitage des acteurs du cloud
  - *« Merci les gars d'avoir fait la R&D pour nous, on va packager tout ça pour en faire un service payant !!!»*
  - ex ?
    - ElasticSearch sur AWS (merci Elastic)
    - Redis sur AWS (merci Redis Lab)
    - Hadoop sur AWS, Azure & GCP (merci Hortonworks, Cloudera & MapR)
    - Glue sur AWS, Spark sur GCP (merci DataBricks)
  - le cas à part de Cassandra
    - AWS a déjà DynamoDB
    - Google a déjà Big Table/Spanner
    - Microsoft … a récemment CosmoDB
    - Oracle cloud humm …

Réaction à la menace du « Cloud »

- Datastax en 2016 se tourne vers le modèle propriétaire, changement radical
  - probablement sous la pression des actionnaires pour + de rentabilité
  - aussi également à cause du clash énorme avec la Fondation Apache (plus de détails sordides à la pause en aparté)

- Paul Dix, CTO d'InfluxData a fait un talk « *The Open Source Business Model is Under Siege* »
  - Idée: open-core == fonctionnalités payante

- Elastic
  - licence spéciale X-Pack interdisant l'utilisation sur du Cloud

- Redis Lab
  - licence « common clause »

Extrait de la licence Common Clause

The Software is provided to you by the Licensor under the License, as defined below, subject to the following condition. **Without limiting other conditions in the License**, the grant of rights under the License **will not include, and the License does not grant to you, the right to Sell the Software**.

For purposes of the foregoing, “Sell” means practicing any or all of the rights granted to you under the License to provide to third parties, **for a fee or other consideration (including without limitation fees for hosting or consulting/ support services related to the Software), a product or service whose value derives, entirely or substantially, from the functionality of the Software**. Any license notice or attribution required by the License must also include this Commons Cause License Condition notice.

Business models du financement participatif

- autant provisionner des pates toute de suite …
- personne ne veut payer !!!!!
- ère du tout gratuit
  - téléchargement film/série
  - musique gratuite sur Spotify/Youtube
  - livres gratuits en pdf
  - jeux crackés (de moins en moins cela dit)
  - Warez
- quelques exemples près de nous
  - qui paie Ippon Technologies (consulting ou autre) pour JHipster ???
  - financement des Cast Codeurs par Patreon

L’éternel débat OSS vs Propriétaire

Le pour et contre: OSS

- pour
  - « gratuit »
  - code ouvert/auditable
  - QA par la communauté sur des scénarios à la marge
  - ~~contribution de la communauté~~

- contre
  - attitude je-m'en-fout-tiste de 90% des utilisateurs
  - TCO = total cost of ownership souvent sous-estimé
    - payer la licence ou payer des experts pour opérer la techno ?
    - coût de la maintenance de la solution (maj, évolution technologique)
  - dominé par les gros acteurs dont l'intérêt diverge avec celui de 90% des utilisateurs (cf clash récent sur la gouvernance de Go …)

Le pour et contre: propriétaire

- pour
  - support et SLA garanti
  - fonctionnalités « premium »
  - mises à jour fréquente
  - time to market plus rapide (à condition de maîtriser la techno)

- contre
  - coût modéré à élevé (combien coûte une équipe de bon devs en France ou à la Silicon Valley ?)
  - code source fermé parfois
  - risque de se lier, de ne plus pouvoir se désengager

OSS vs propriétaire

Merci !

@doanduyhai

doanduyhai@gmail.com