

# TP Cassandra

12 février 2013

Philippe Girolami  
Nicolas Romanetti

# Objectifs TP Cassandra

- Installer plusieurs instances Cassandra pour former un cluster
- Insérer/Consulter des données depuis la console
- Simuler des pannes
- Ajouter/Retirer un noeud
- Questions/Réponses

# Installation Cassandra

- **Télécharger Cassandra 1.1.9**  
<http://apache.crihan.fr/dist/cassandra/1.1.9/apache-cassandra-1.1.9-bin.tar.gz>
- Installer Cassandra sur son poste (pas besoin d'être root)
  - ▶ 'untarer' dans son home directory
- Ouvrir dans un éditeur le fichier **conf/cassandra.yaml**
  - ▶ remplacer /var/lib/cassandra par <path absolu vers votre home dir>/var\_lib\_cassandra  
(ne pas utiliser ~ => pas reconnu)

# Installation Cassandra

- **Calculer votre token** sachant qu'il y a N machines (N connu le jour du TP)  
Utiliser le programme Java (slide suivant) ou le service en ligne suivant:  
<http://blog.milford.io/cassandra-token-calculator/>
- ▶ Attribuer un index (on part de 0) à chaque machine
- ▶ Chacun calcule son token à partir de son index
- ▶ Copy paste du token dans le fichier conf/cassandra.yaml  
initial\_token: votre\_token\_ici
- ▶ Relisez-vous!
- **Conf IP**
- ▶ listen\_address: <votre ip>
- ▶ rpc\_address: <votre ip>
- Choisir 2 points de ralliements (**les seeds**) communs à tous
- ▶ - seeds: "une\_ip,une\_autre\_ip"

# Calculer le token en Java

```
// STC.java
// Simple Token Calculator
// compile: javac STC.java
// run: java STC nbNodes index

import java.math.BigInteger;
public class STC {
    public static void main(String args[]) {
        int nbNodes = Integer.parseInt(args[0]);
        int index = Integer.parseInt(args[1]);
        BigInteger token = new BigInteger("2");
        token = token.pow(127);
        token = token.multiply(new BigInteger("" + index));
        token = token.divide(new BigInteger("" + nbNodes));
        System.out.println("initial_token: " + token);
    }
}
```

# Installation Cassandra

- ouvrir le fichier `conf/log4j-server.properties` et remplacer
  - ▶ `log4j.rootLogger=INFO,stdout,R`  
*par*  
`log4j.rootLogger=DEBUG,stdout`
- Démarrer Cassandra (-f pour 'foreground')  
[ATTENDRE QUE TOUT LE MONDE SOIT PRÊT]
  - ▶ On vérifie chaque poste
  - ▶ `bin/cassandra -f`

# Nodetool

- Utilitaire d'administration pour
  - ▶ obtenir des statistiques
  - ▶ manager les instances Cassandra
- Se connecte via JMX (par défaut port 7199) à une instance Cassandra
  - ▶ `ex: bin/nodetool -h localhost ring;`
- On vérifie que tout le monde est présent dans le ring

# Client Cassandra

## ◎ Utilisation de `cassandra-cli`

[http://www.datastax.com/docs/1.1/dml/using\\_cli](http://www.datastax.com/docs/1.1/dml/using_cli)

### ▶ Client cassandra pour

- créer/updater le schema
- écrire/lire des données

### ▶ Se connecte à Cassandra via Thrift (port 9160 par défaut)

- `bin/cassandra-cli -h localhost`



# Création du schéma

- **Une seule personne** crée le keyspace que nous allons utiliser à l'aide de **cassandra-cli**

- ▶ `create keyspace ks with placement_strategy = 'SimpleStrategy' and strategy_options = {replication_factor:3};`
- ▶ `use ks;`
- ▶ `create column family user with key_validation_class='AsciiType' and comparator='AsciiType' and default_validation_class='AsciiType';`

# Insertion de données

- `bin/cassandra-cli -h localhost`
- Se placer dans le Keyspace ks
  - ▶ `use ks;`
- Mode ascii pour nos commandes:
  - ▶ `assume user keys as ascii;`
  - ▶ `assume user comparator as ascii;`
  - ▶ `assume user validator as ascii;`
- Insérer des données
  - ▶ `set user['toto']['firstname']='Antoine';`
- Récupérer une donnée
  - ▶ `get user['toto'];`

# Insertion de données

- Insérer une colonne avec un TTL 30sec

- ▶ `set user['toto']['password']= 'xyz' with ttl=10;`
- ▶ `list user;`
- ▶ la colonne disparaît après 30sec

- Counter

- ▶ (à faire par une seule personne)  
`CREATE COLUMN FAMILY hits WITH  
default_validation_class=CounterColumnType AND  
key_validation_class=UTF8Type AND comparator=UTF8Type;`
- ▶ Tous le monde peut incrémenter...  
`incr hits['www.twitter.com']['home'] by 1;`

- Secondary index sur «country» (à faire par une seule personne)

- ▶ (à faire par une seule personne)  
`UPDATE COLUMN FAMILY user WITH column_metadata=[{column_name:  
country, validation_class: AsciiType, index_type: KEYS}];`
- ▶ Tous le monde peut requêter..  
`GET users WHERE country = 'FR';`

# Se familiariser

- Ajoutez une deuxième colonne
- Retirez une colonne
- Listez les rows de la CF User
- Supprimez une row

**Utilisez** `help;` ou `help <cmd>;`

# JConsole / JMX

- ◎ JMX permet de monitorer et d'interagir avec la JVM
  - ▶ au niveau interne
  - ▶ au niveau applicatif
- ◎ JConsole est un client JMX fournit avec la JVM
  - ▶ `jconsole`

# Ecriture et observation du ring

- Que se passe-t-il si un noeud est down?
- Changer le consistency level de cassandra-cli:
  - ▶ `consistencylevel as ALL;`
- Faire tomber un ou plusieurs noeud, que se passe-t-il lors de l'écriture?
- Repasser à CL `ONE`, que se passe-t-il lors de l'écriture?
- Relancer les noeuds down, qu'observez-vous dans les logs?

# Node Repair

- © Rappel

- ▶ Node repair permet de re-synchroniser les éventuels writes infructueux, notamment les 'delete'
- ▶ A lancer sur chaque noeud au moins une fois par `grace_period_seconds`
  - `nodetool -h <host> repair`
- ▶ Attention: ne pas lancer cette commande sur tous les noeuds en même temps: très gourmand en CPU / accès disque

# Remplacer un noeud défaillant

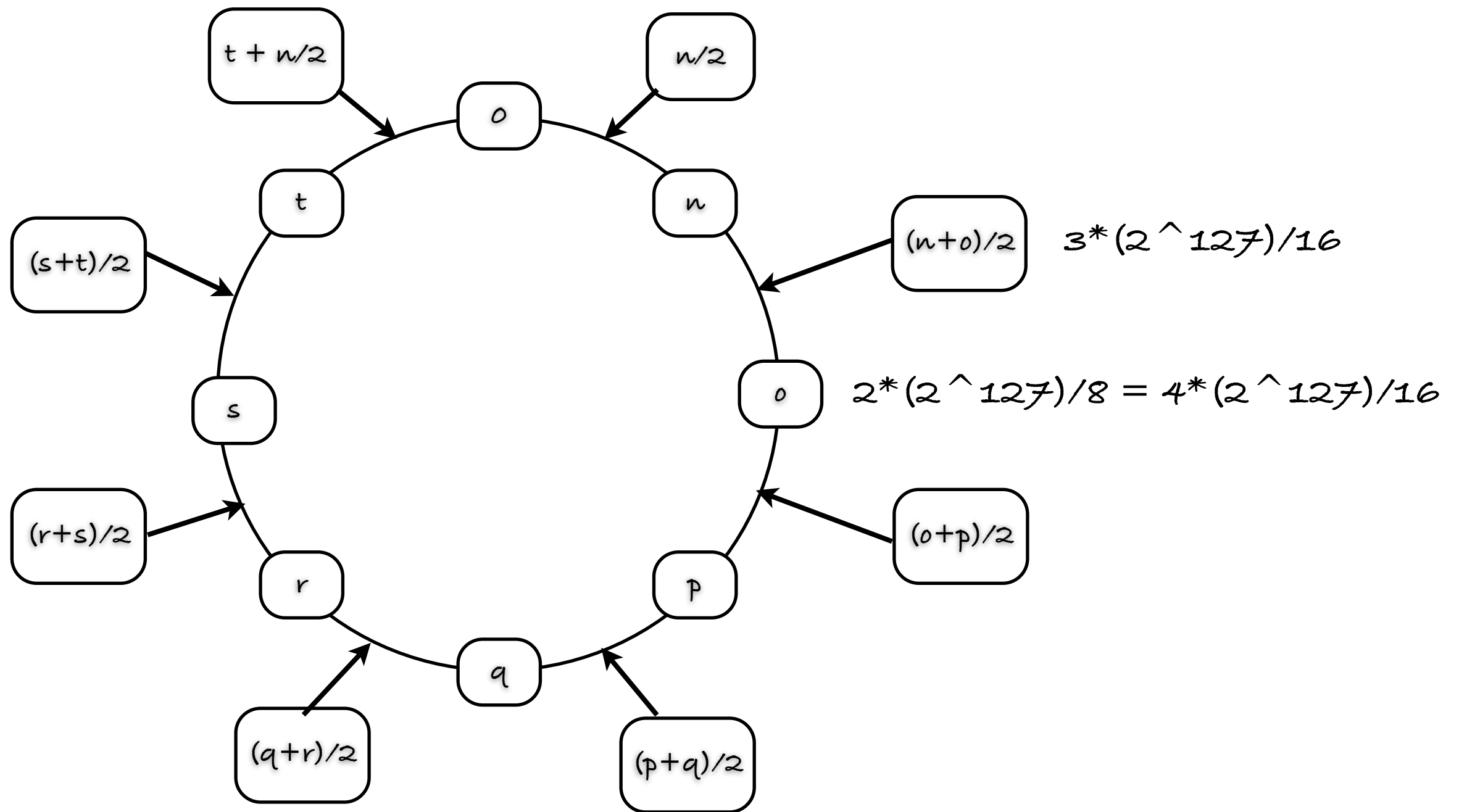
- Installer Cassandra sur le noeud
- Reconfigurer le noeud comme il était (même token)
  - ▶ IL NE DOIT AVOIR AUCUNE DONNEE
- Lancer cassandra avec l'option: `-Dcassandra.replace_token`
  - ▶ `bin/cassandra -f -Dcassandra.replace_token <token>`
- Etre patient, le noeud va récupérer les données...
- Une fois les données récupérées, le noeud accepte les write
- Par précaution, lancez un `nodetool repair` juste après



# Ajout de noeuds

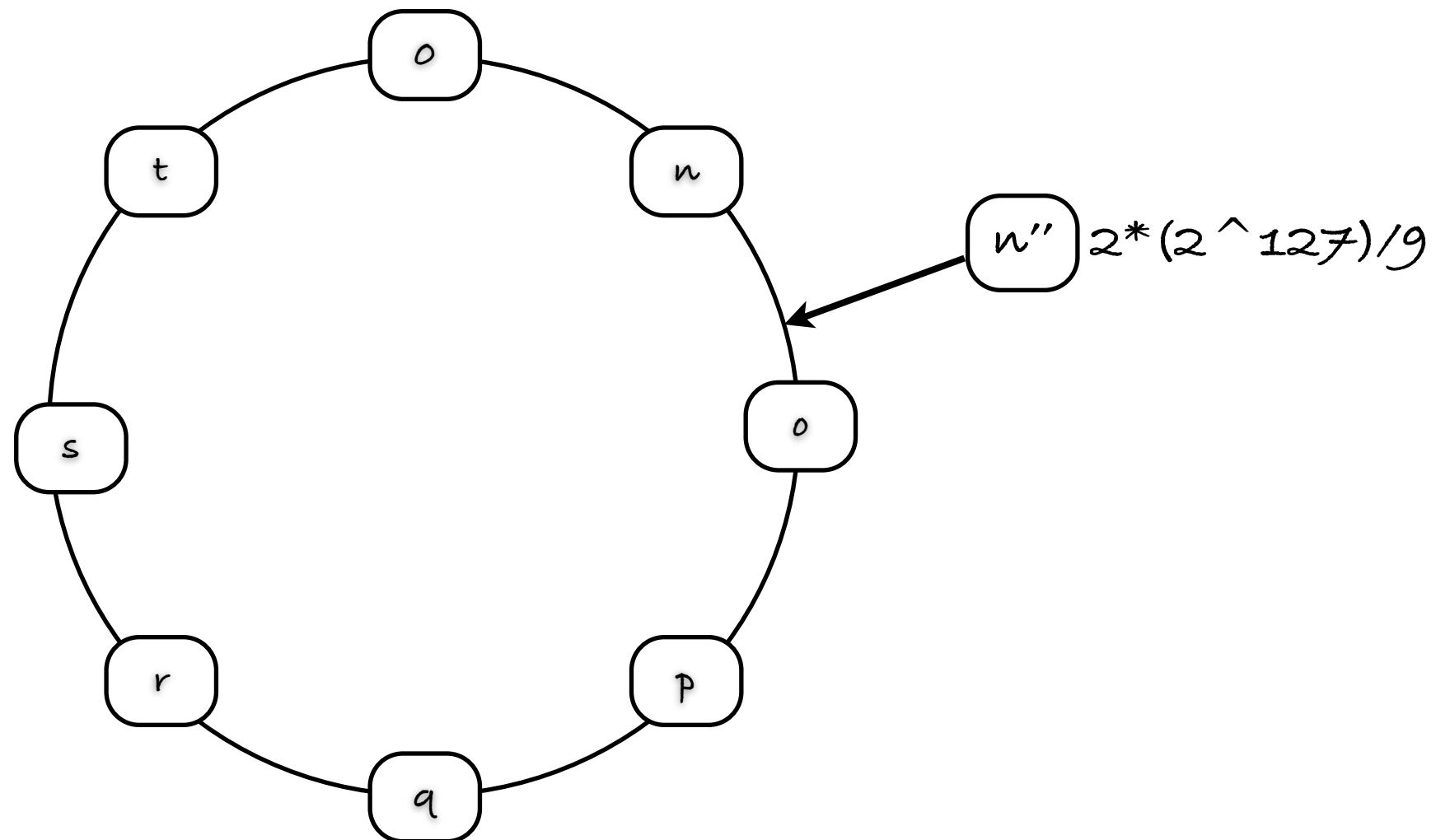
- ◎ Doubler la taille du cluster
  - ▶ Ajoute 1 noeud entre chaque token
  - ▶ Conséquence: divise par 2 la charge de chaque noeud
  - ▶ Inutile de re-calculer les tokens existants

# Doubler la taille du cluster



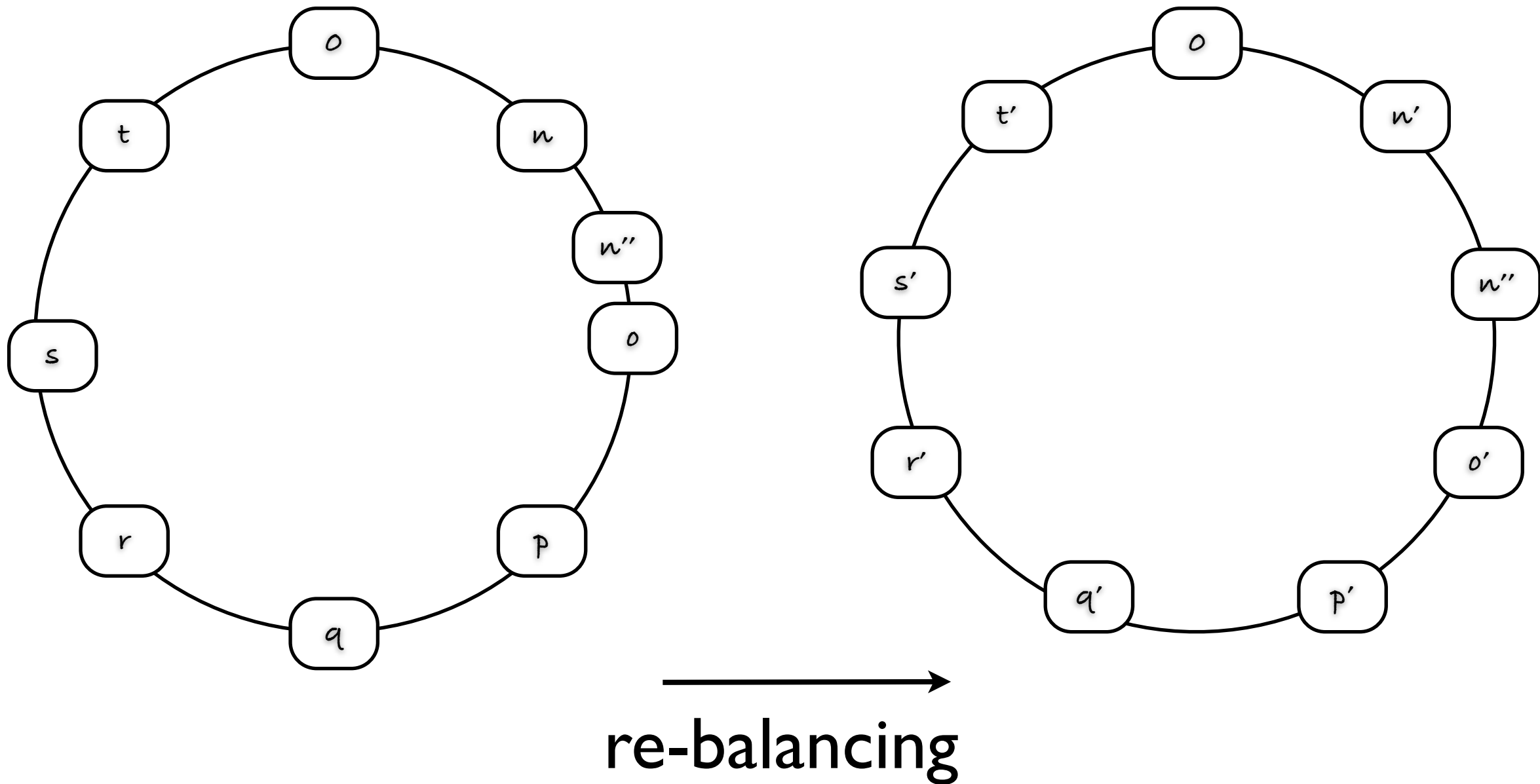
Les anciens noeuds conservent leur token

# Ajouter 1 noeud



On crée un déséquilibre...

# Ajouter 1 noeud



On «déplace» les anciens noeuds pour retrouver l'équilibre

# Ajouter un noeud

- Ajout d'un seul noeud
  - ▶ Possible en attribuant un token manuellement ou automatiquement
  - ▶ Conséquence: rompt la répartition uniforme
    - Solution: re-calculer et re-assigner un token aux anciens noeuds
- Réassigner un token
  - ▶ Pour chaque ancien noeud:
    - **Editer** `conf/cassandra.yaml` pour mettre à jour le `initial_token`
    - **puis lancer**  
`nodetool -h <host> move <newToken>`
    - **puis nettoyer les données n'appartenant plus au noeud**  
`nodetool -h <host> cleanup`
    - `update keyspace ks with strategy_options = {replication_factor : 3};`

# Supprimer un noeud

- Supprimer un noeud déjà hors service

- ▶ lancer

- ```
nodetool -h <autreHost> removetoken <token>
```

- où token= token du noeud à supprimer

- ▶ Les réplicas streament des données vers le noeud choisi par Cassandra pour prendre en charge l'intervalle de token

- Supprimer un noeud encore en service

- ▶ lancer

- ```
nodetool -h <hostASupprimer> decommision
```

- ▶ Ce noeud va alors streamer ses données vers le noeud choisi par Cassandra...

Tout est fait par Cassandra pour que le Replication Factor soit bien respecté

attention au déséquilibre...

# Monitoring

- Monitorer Cassandra

- ▶ nodetool help
- ▶ jconsole (jmx console)
- ▶ Les logs (voir log4-server.properties)

- OpsCenter

- ▶ produit datastax
- ▶ demande l'installation d'un agent sur chaque noeud


# Architecture Interne & Monitoring

## ● Monitoring des «stages» cassandra:

`nodetool tpstats`

(‘tp’ pour thread pool)

(1) => 1 thread seulement

Pool Name	Active	Pending	Completed	Blocked	All time blocked
ReadStage	0	0	1	0	0
RequestResponseStage	0	0	0	0	0
MutationStage	0	0	20	0	0
ReadRepairStage (1)	0	0	0	0	0
ReplicateOnWriteStage	0	0	0	0	0
GossipStage (1)	0	0	86919	0	0
AntiEntropyStage (1)	0	0	0	0	0
MigrationStage (1)	0	0	0	0	0
MemtablePostFlusher	0	0	4	0	0
StreamStage	0	0	0	0	0
FlushWriter	0	0	4	0	0
MiscStage (1)	0	0	0	0	0
InternalResponseStage	0	0	0	0	0
HintedHandoff	0	0	13	0	0
Message type	<b>Dropped</b>				
RANGE_SLICE	0	 Messages expirés sont «drippés»			
READ_REPAIR	0				
BINARY	0				
READ	0				
MUTATION	0				
REQUEST_RESPONSE	0				

Système sain: ‘Active’, ‘Pending’ & Dropped à 0 ou très proche de 0



# Statistique par Column Family

## ● Column Family

► `nodetool -h <host> cfstats`

```
Column Family: test
SSTable count: 1
Space used (live): 5218
Space used (total): 5218
Number of Keys (estimate): 128
Memtable Columns Count: 0
Memtable Data Size: 0
Memtable Switch Count: 0
Read Count: 0
Read Latency: NaN ms.
Write Count: 0
Write Latency: NaN ms.
Pending Tasks: 0
Bloom Filter False Postives: 0
Bloom Filter False Ratio: 0,00000
Bloom Filter Space Used: 496
Key cache capacity: 200000
Key cache size: 0
Key cache hit rate: NaN
Row cache: disabled
Compacted row minimum size: 87
Compacted row maximum size: 103
Compacted row mean size: 103
```