Introduction à

riak

Dominic Williams

Extreme Forge

Erlang Factory Lite Paris 2012

Riak

- Produit de Basho
- Base clé/valeur inspirée de Dynamo (Amazon)
  - Avec quelques supplements: recherche, MapReduce, liens, 2i, hooks pre/post-commit, back-end de stockage modifiable, interface HTTP et binaire (protocal buffers)
- Ecrit en Erlang avec un peu de C/C++
- Open Source (licence Apache 2)

Les objectifs de Riak

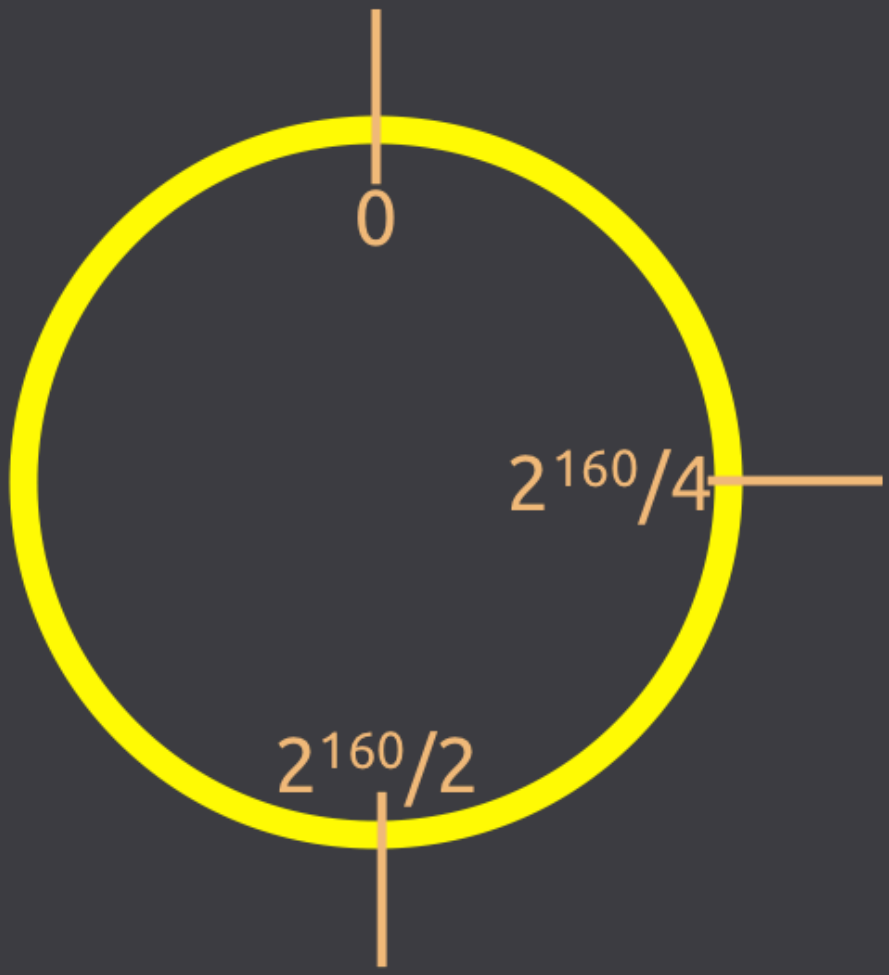
- Haute disponibilité
- Faible latence
- Scalabilité horizontale
- Tolérance aux pannes
- Simplicité d'exploitation
- Prédictibilité
- Priorité à la disponibilité
- Possibilité de régler pour priorité à la cohérence

ACID, CAP et BASE

- ***A***tomicity, ***C***onsistency, ***I***solation, ***D***urability
  - Caractéristiques des bases de données traditionnelles
  - Obstacles de scalabilité et de distribution WAN
- ***C***onsistency, ***A***vailability, ***P***artition tolerance
  - Théorème de Brewster/Gilbert/Lynch
  - “On ne peut pas tout avoir tout le temps”
- ***B***asically ***A***vailable, ***S***oft-state, ***E***ventually consistent
  - Option adoptée par les géants du Web et la plupart des bases NoSQL

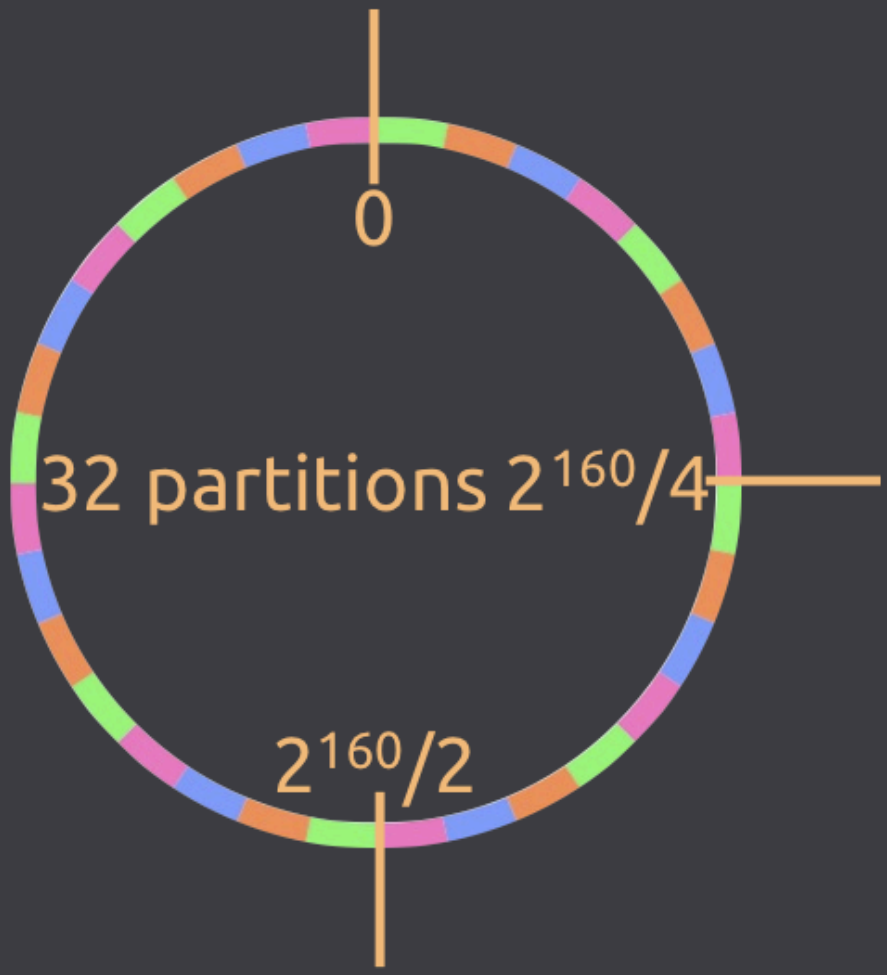
Consistent Hashing

- 160-bit integer keyspace



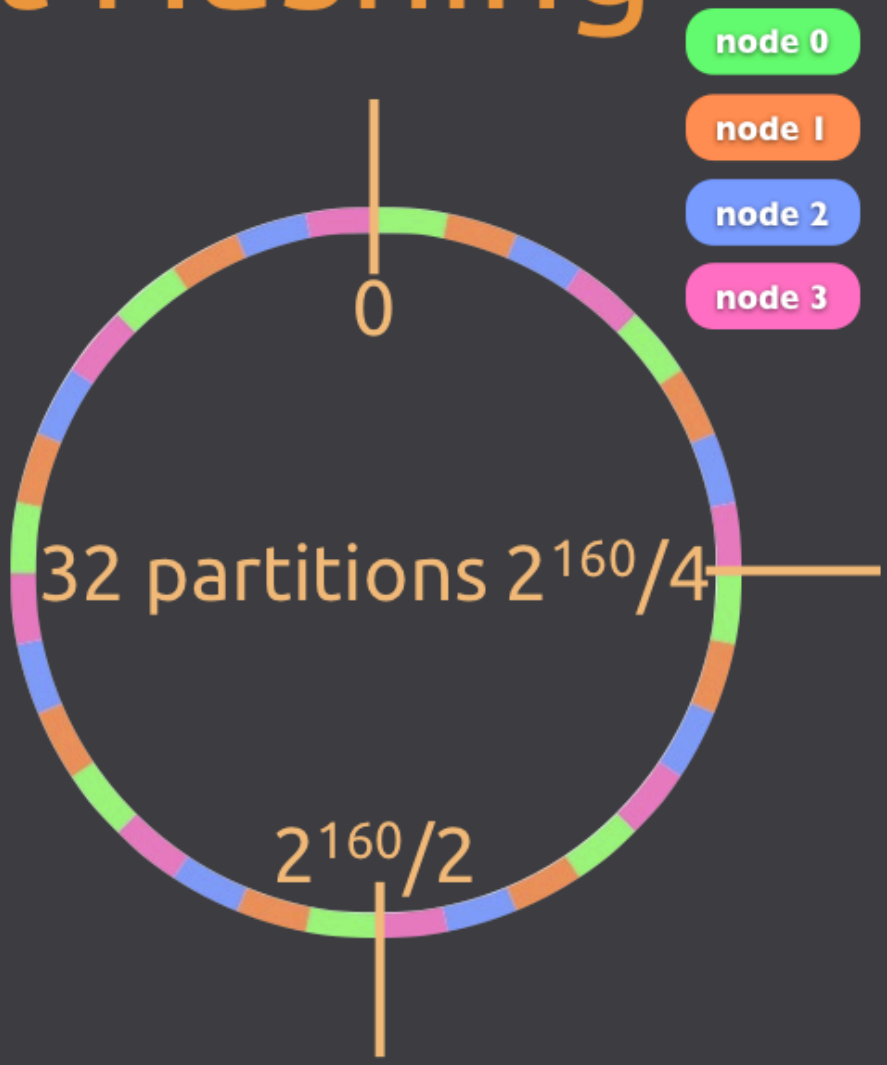
Consistent Hashing

- 160-bit integer keyspace
- divided into fixed number of evenly-sized partitions



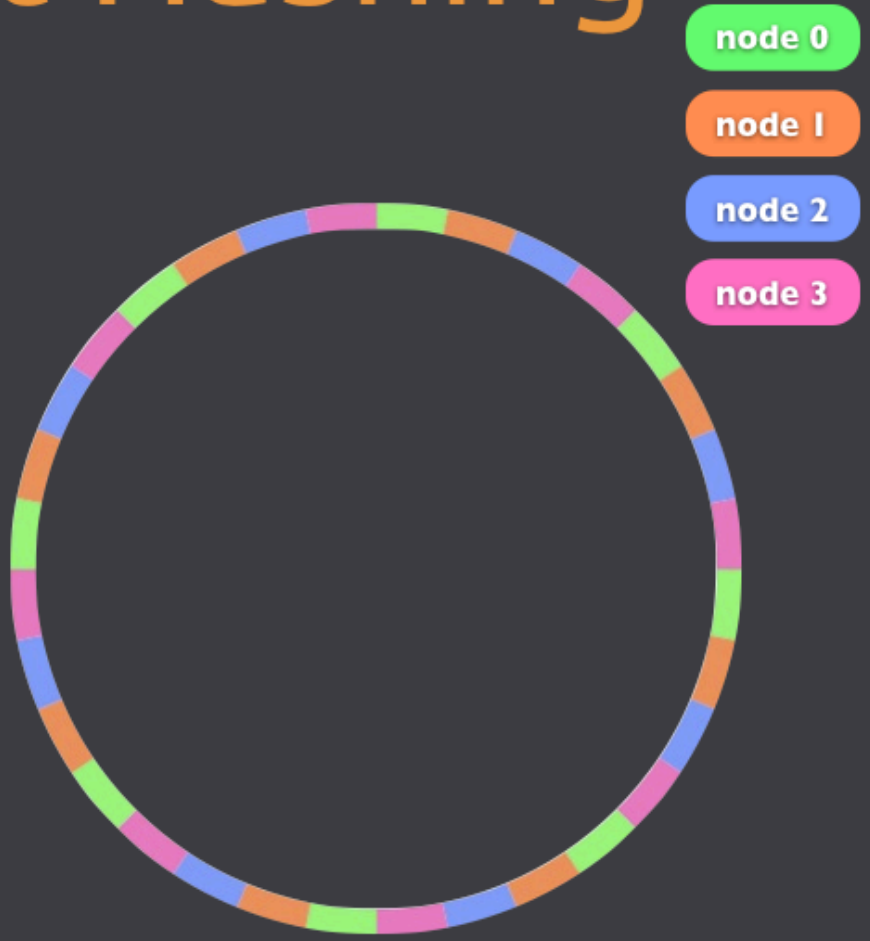
Consistent Hashing

- 160-bit integer keyspace
- divided into fixed number of evenly-sized partitions
- partitions are claimed by nodes in the cluster



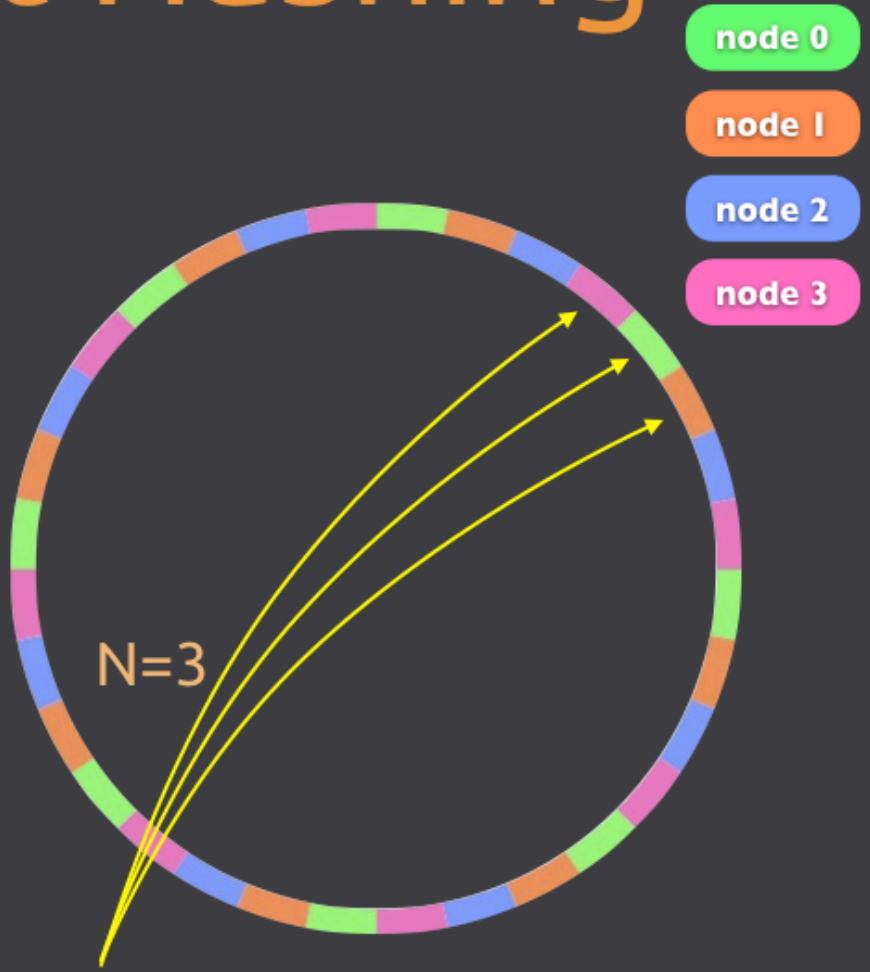
Consistent Hashing

- 160-bit integer keyspace
- divided into fixed number of evenly-sized partitions
- partitions are claimed by nodes in the cluster
- replicas go to the N partitions following the key



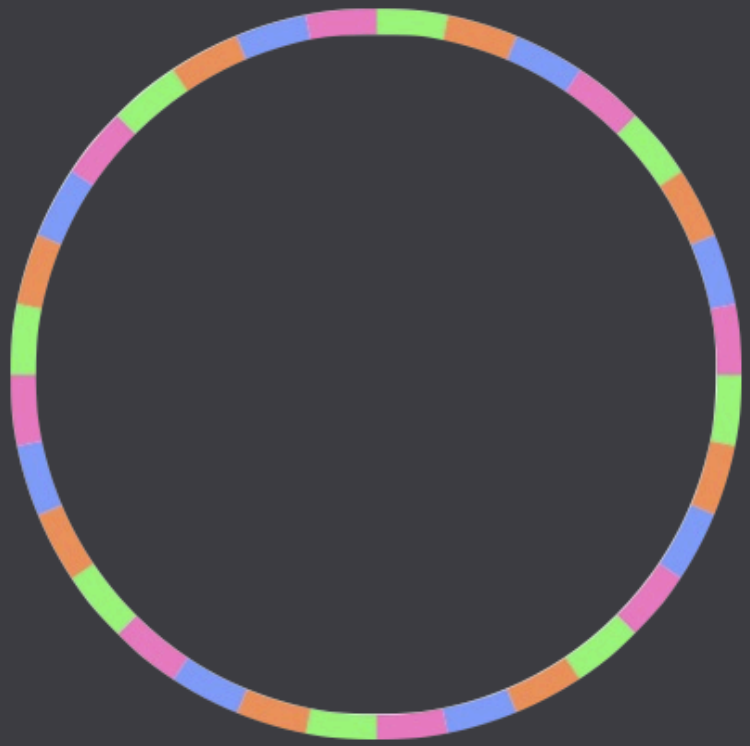
Consistent Hashing

- 160-bit integer keyspace
- divided into fixed number of evenly-sized partitions
- partitions are claimed by nodes in the cluster
- replicas go to the N partitions following the key



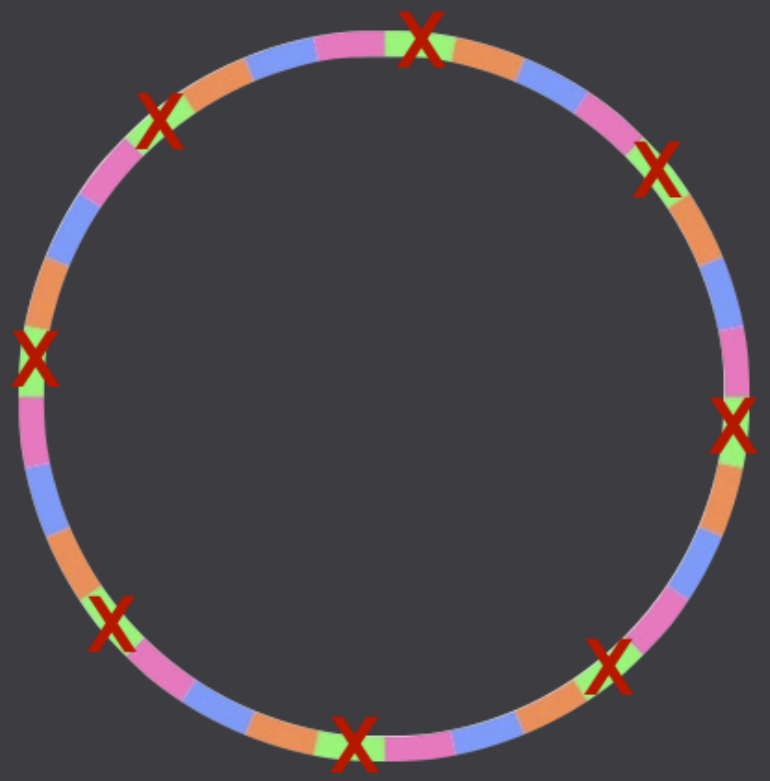
hash(“meetups/nycdevops”)

Disaster Scenario



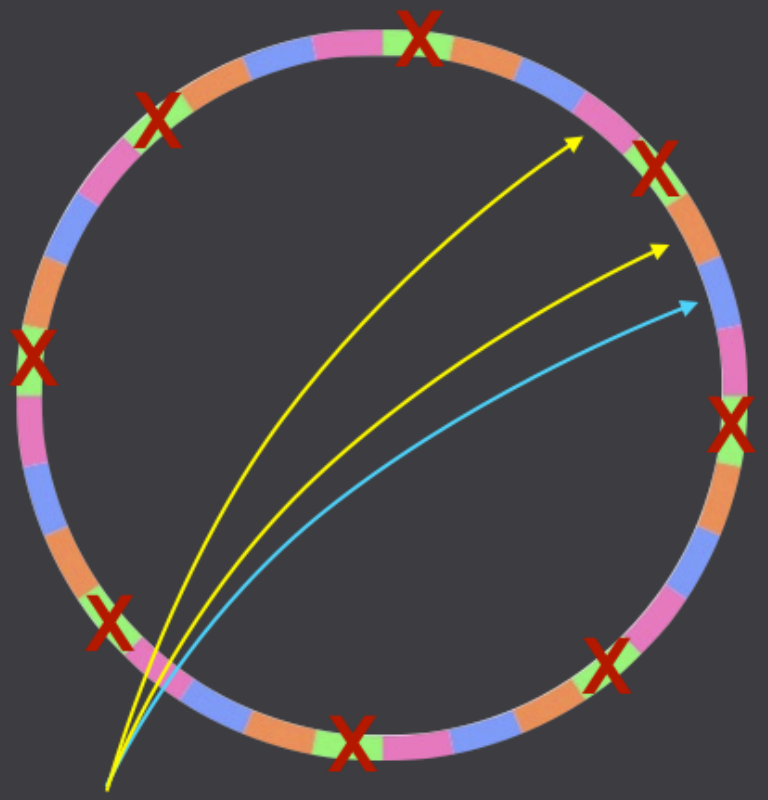
Disaster Scenario

- node fails



Disaster Scenario

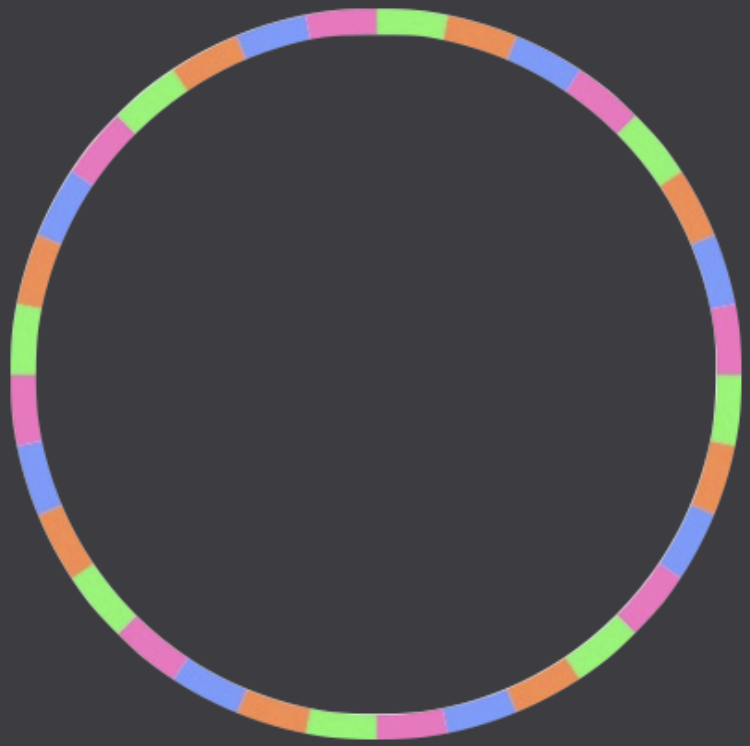
- node fails
- requests go to fallback



hash(“meetups/nycdevops”)

Disaster Scenario

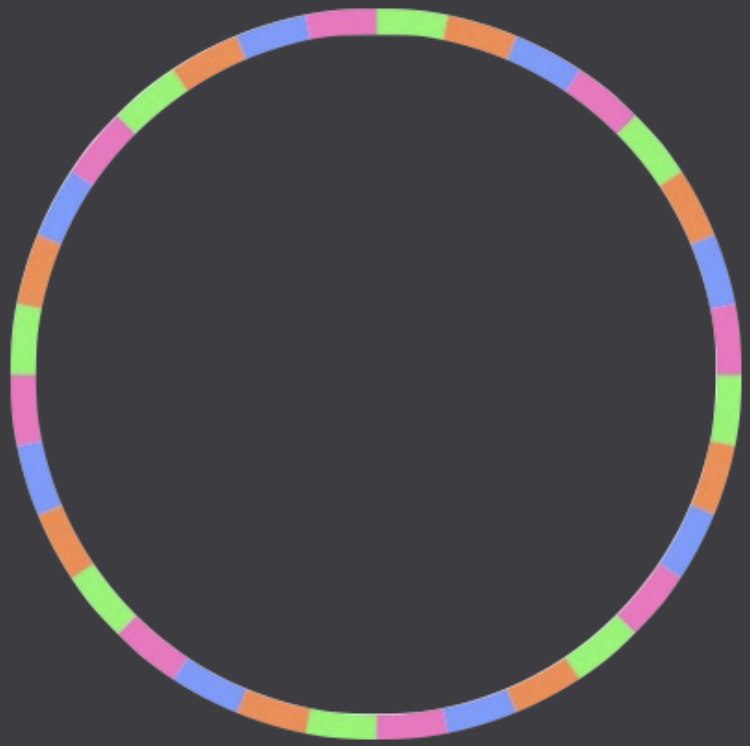
- node fails
- requests go to fallback
- node comes back



hash(“meetups/nycdevops”)

Disaster Scenario

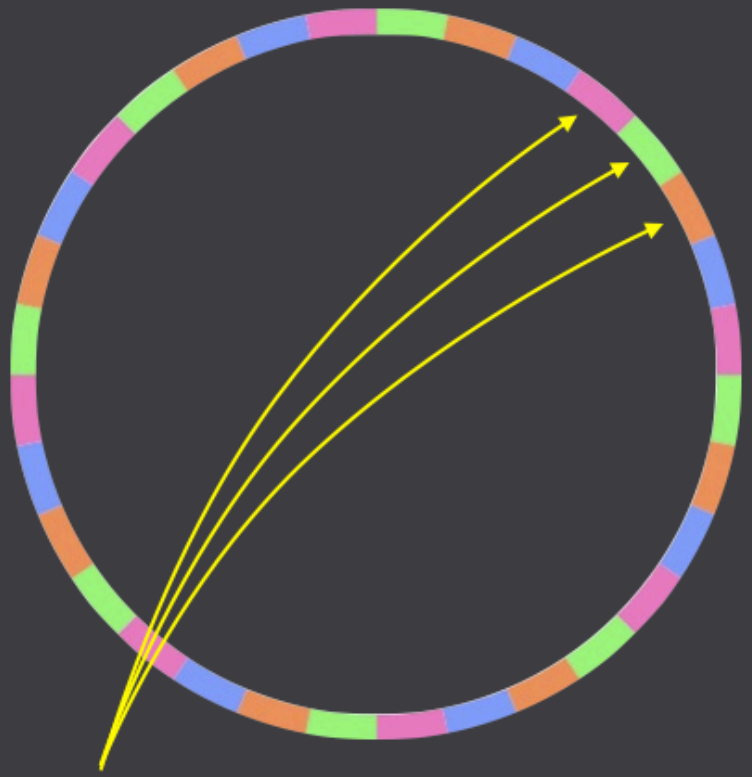
- node fails
- requests go to fallback
- node comes back
- “Handoff” - data returns to recovered node



hash(“meetups/nycdevops”)

Disaster Scenario

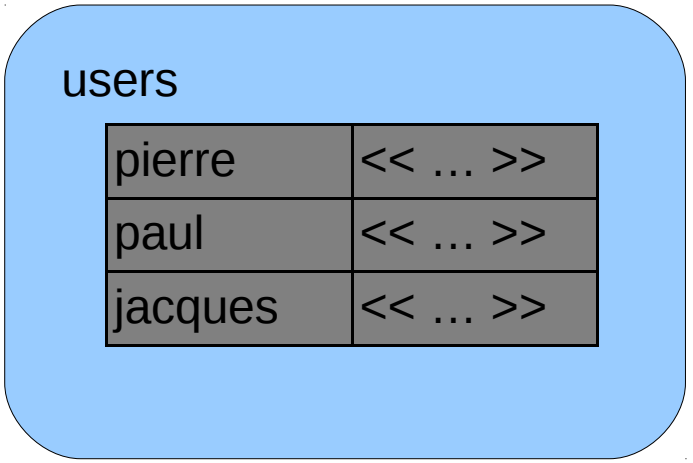
- node fails
- requests go to fallback
- node comes back
- “Handoff” - data returns to recovered node
- normal operations resume

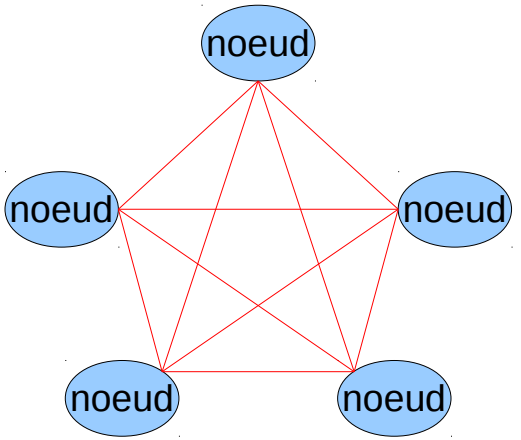


hash(“meetups/nycdevops”)

A quoi ça ressemble pour un développeur

- HTTP ou protobuf
- Lib cliente Erlang, JS etc.
- Seau → Clé → Valeur
- GET/PUT/DELETE
- MapReduce
- Recherche texte
- Indices secondaires
- Pas de transactions
- Gestion applicative des incohérences





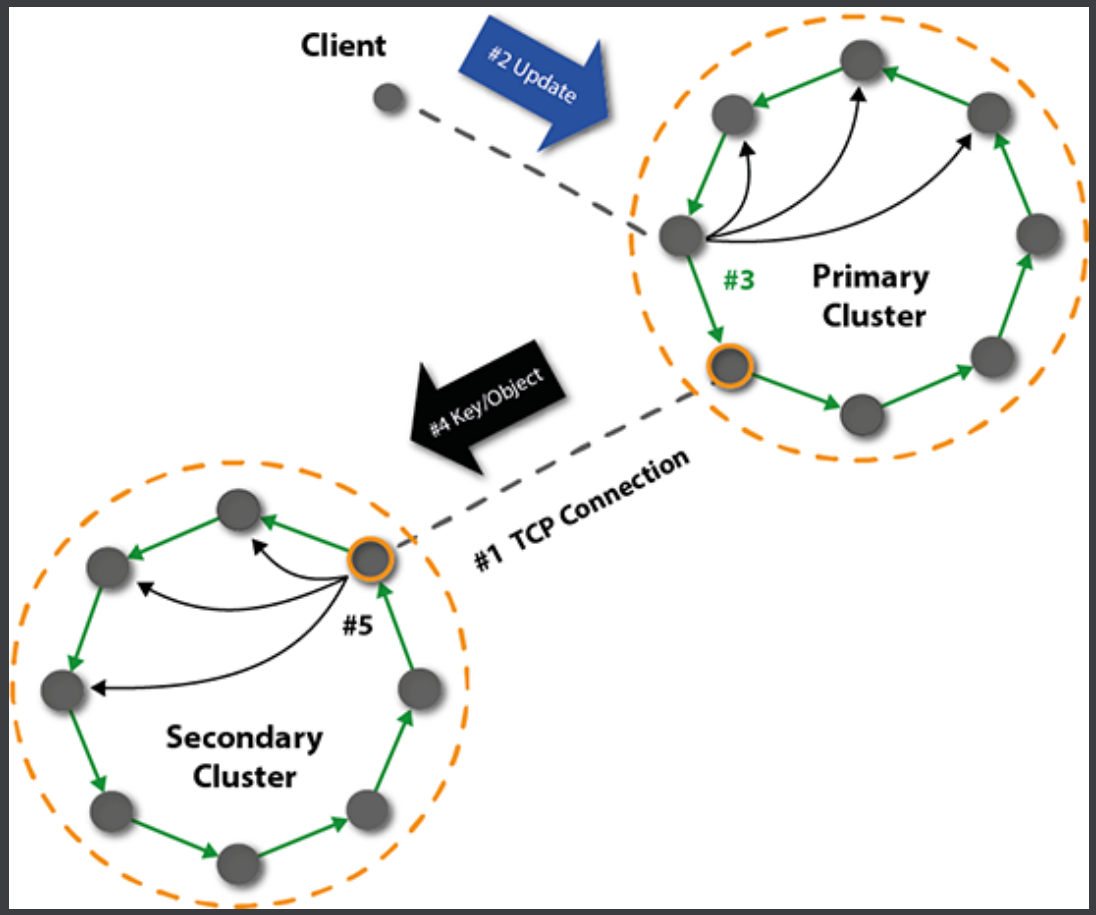
Réorganisation automatique

- Les noeuds discutent entre eux:
  - “gossip protocol”
  - Traffic réseau

- Lorsque la topologie du cluster change

  - Détection automatique du changement (nouveau noeud, perte d'un noeud)
  - Réorganisation automatiquement les données
  - En tâche de fond (sans interruption de service)

Riak Enterprise: Real-Time Sync



Cohérence vs. disponibilité

Paramètre	Description	Défaut
n_val	Nombre de copies à stocker (réglable par seau)	3
w	Nombre de copies requises pour qu'une écriture réussisse	2
r	Nombre de copies disponibles pour qu'une lecture réussisse	2

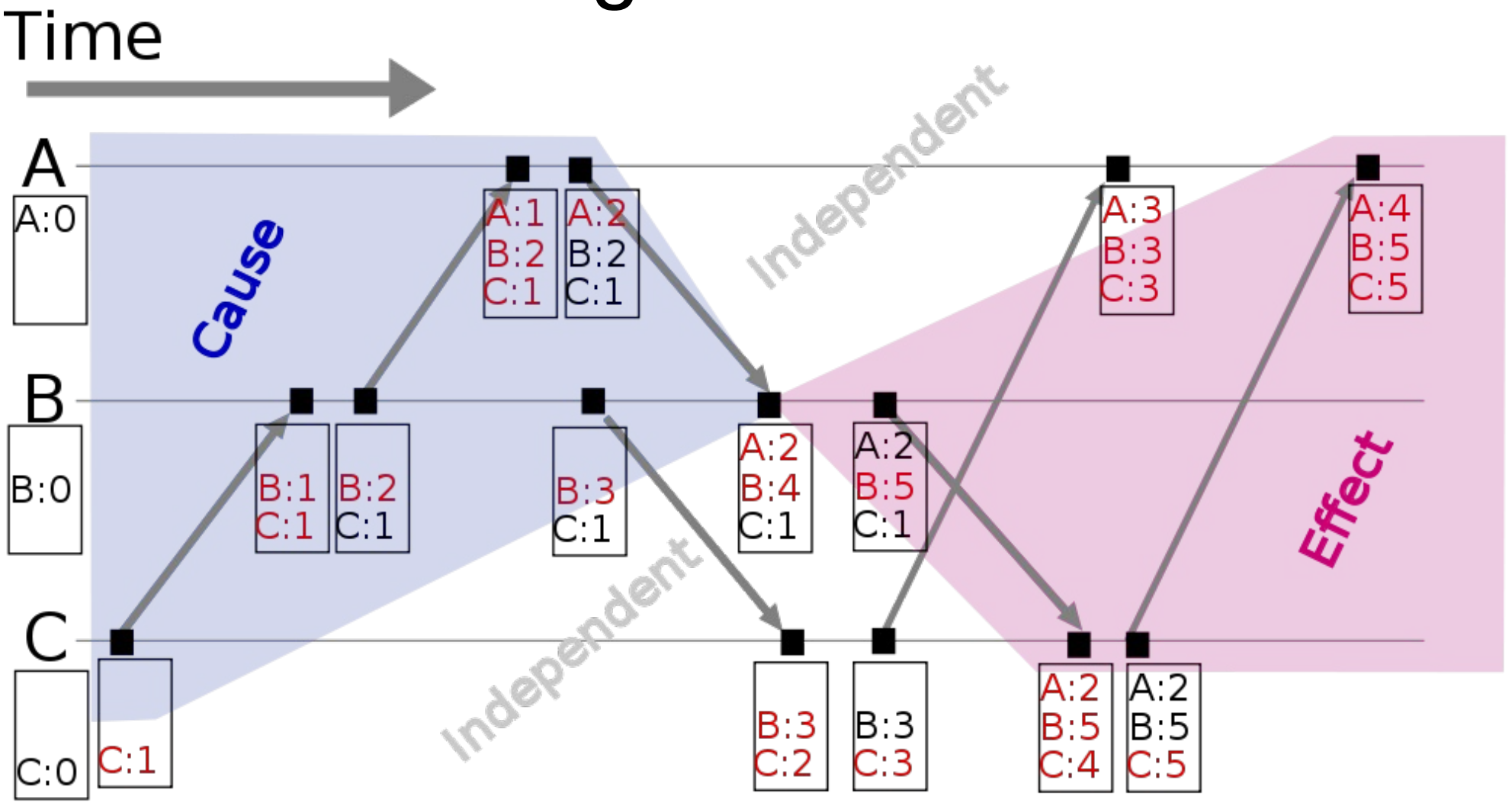
Cohérence “à terme”

- En fonction des paramètres n_val, w et r:
  - Toutes les copies d'une valeur peuvent ne pas êtres cohérentes tout de suite
  - Elles le seront “à terme”
  - Mais entre-temps, deux clients peuvent obtenir une valeur différente pour une même clé...

Gestion des incohérences

- Utilisation d'horloges vectorielles
- En général, résolution automatique
- Parfois (en cas de partitionnement temporaire ayant conduit à une résolution impossible):
  - L'application se retrouve avec deux versions possibles de la valeur
  - L'application doit décider elle-même comment résoudre le conflit

Horloge vectorielle



Exemples de résolution

- La dernière écriture gagne
- Stocker les opérations (si commutatives)
- Privilégier une valeur en fonction d'une règle métier:
  - Nombre d'articles dans un panier: prendre le plus grand
- Donner priorité à la valeur fixée par certains clients
- Demander à l'utilisateur:
  - e.g. “'Nous sommes désolés, nous avons deux version pour n° de tél., pouvez-vous choisir la bonne?”

Atomicité mais pas de transactions

- L'écriture d'une valeur est atomique (ça réussit sur w noeuds, ou ça échoue proprement partout)
- Pas de transactions, donc pas possible d'écrire dans plusieurs clés de façons atomiques
- Solution simple: utiliser des valeurs complexe (e.g. un record avec tous les champs qui doivent rester cohérents)
- Mais ça devient impossible de dénormaliser la base pour optimiser les lectures et les recherches...

Retours d'expérience 1/2

- Installation facile (si bonne version OTP etc)
- Bonne documentation et bon support Basho
- MIG:
  - Après un premier essai sur produit ambitieux, ont gardé Riak pour 2ème grand produit
  - Très contents du support, des perfs et de la stabilité
  - Pas mal de travail pour arriver à une cohérence suffisante

Retours d'expérience 2/2

- Corporama:
  - Pas retenu après une tentative sérieuse
    - Contents de la doc et du support
    - problèmes de perfs. et stabilité
    - Riak Search pas encore mature
- Klarna:
  - Bien que très expérimentés avec Mnesia, ont des problèmes de volume et de stabilité
  - sont en train de préparer un passage à Riak
  - Ont développé un système très sophistiqué (Meshup) pour introduire une couche transactionnelle tout en simplifiant l'utilisation...

Démo

Conclusions

- Riak est une bonne solution:
  - Quand le volume de données nécessite >5 serveurs
  - Quand le modèle de données et les requêtes nécessaires le permettent en toute simplicité
- Grande simplicité d'exploitation
- Attention au développement complexe:
  - Pour gérer la cohérence
  - Pour contourner l'absence de transactions
  - Pour faire du Map-Reduce
- Il faut ***essayer et mesurer***, pour chaque besoin

Références

- Basho: overview, wiki et speakerdeck
- *Dynamo: Amazon's Highly Available Key-value Store*, SOSP 2007
- Wikipedia: consistent hashing, vector clock
- *Distributed Algorithms* by Nancy Lynch
- Marcus Kern (MIG) at Erlang Factory 2011
- Jakob Sievers (Klarna) at Erlang Factory 2012
- Nicolas Thauvin (Corporama): personal communication