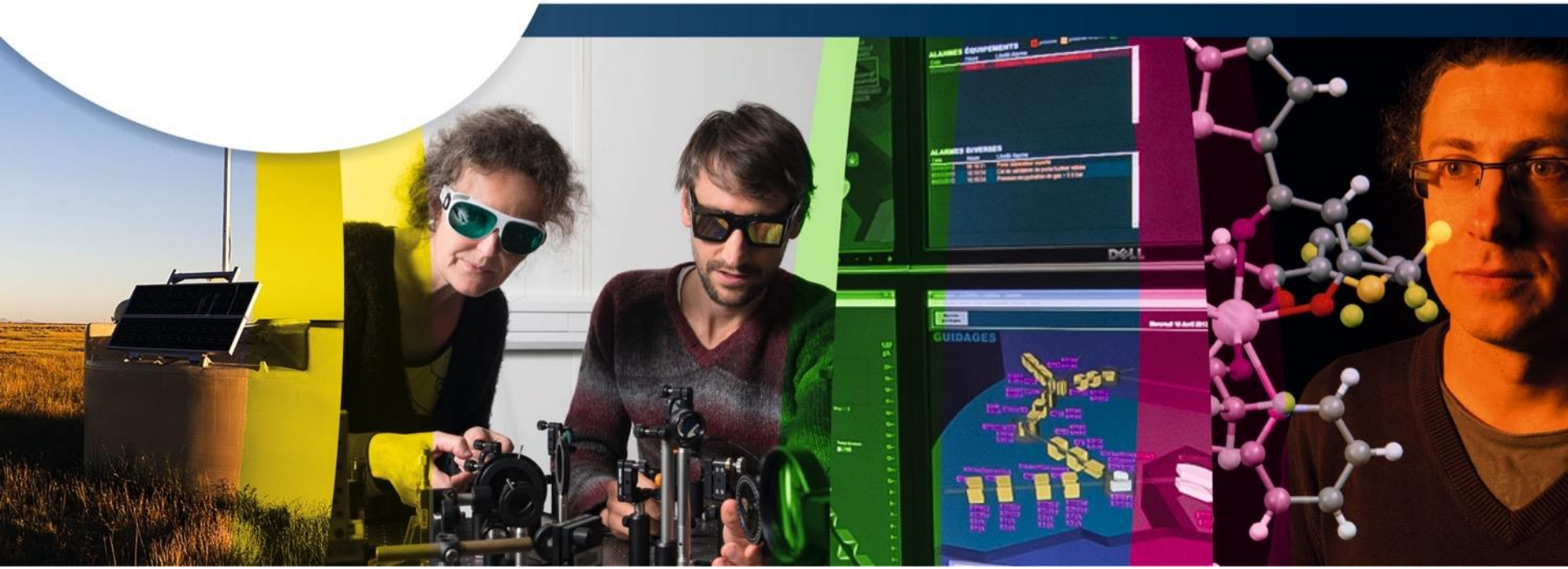
Bases de données Clés-Valeurs avec



www.cnrs.fr

Sommaire

Introduction
Fondamentaux
Travaux pratiques

www.cnrs.fr

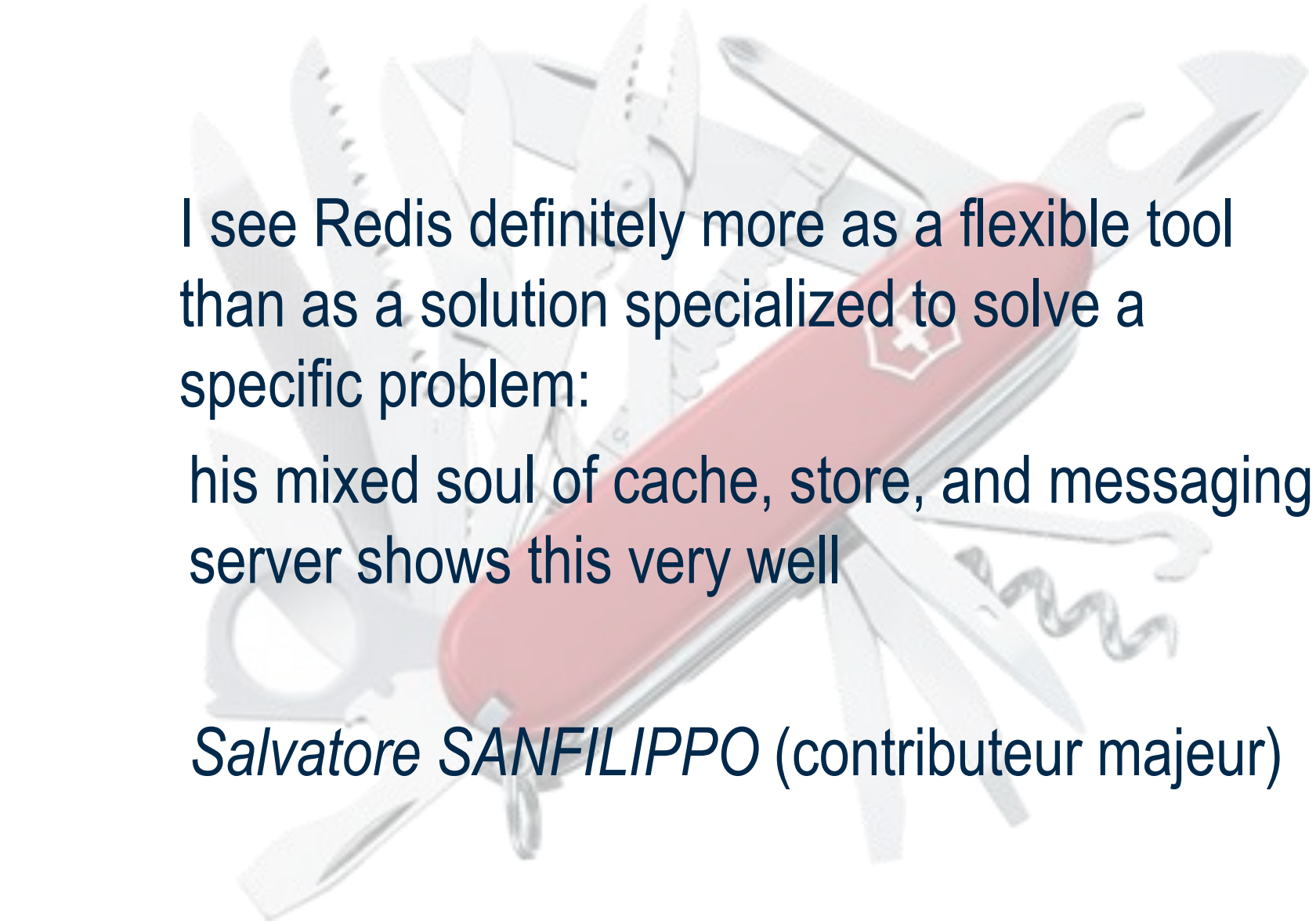
Introduction

Historique

- Créé en 2009

- Créé initialement pour améliorer les performances de l'outil d'analyse des visites de sites Web LLOOGG

Qu’est-ce que Redis ?



I see Redis definitely more as a flexible tool than as a solution specialized to solve a specific problem:

his mixed soul of cache, store, and messaging server shows this very well

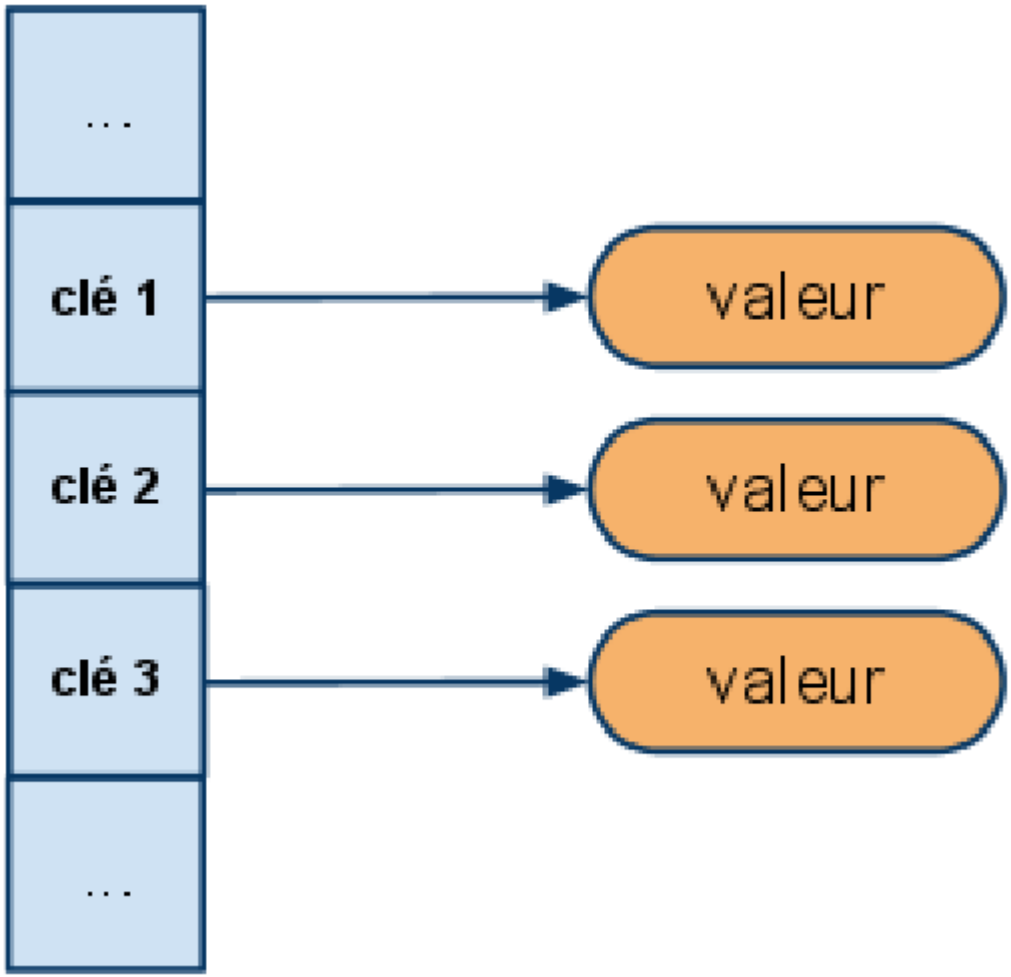
Salvatore SANFILIPPO (contributeur majeur)

Qu’est-ce que Redis ?

REmote DIctionary Server

(serveur de dictionnaire distant)

Fonctionnement



Fonctionnement

Fonctionnement

- 1 base = 1 table

Clés primaires	Valeurs
Loto : Samedi 23 avril 2017	2, 15, 17, 22, 26 - 7
La grande question sur la vie, l'univers et le reste	42
logo cnrs	

http://prodev.cnrs.fr/doku.php?id=nosql2017	`<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr" dir="ltr" class="no-js">`
	
	`<head>`
	`<meta charset="UTF-8" />`
	`<title>nosql2017 [prodev]</title>`
	…
	`<body>`
	…
	`</body> </html>`

Fonctionnement

- Conserve l'intégralité des données en RAM
  - Permet d'obtenir d'excellentes performances en évitant les accès disques
- Utilise le disque pour persister les données
- Réplication maître-esclave
- Support de nombreux types de données
- Chaque commande est une transaction atomique

Fonctionnement

SET la-reponse 42

OK

Fonctionnement

GET la-reponse

42

Performances

- Environ 50 000 opérations (lecture/écriture) par seconde
- Jusqu'à 100 000 opérations par seconde sur EC2

Installation

Construction depuis les sources

- `wget http://download.redis.io/releases/redis-3.2.0.tar.gz`
- `tar xzf redis-3.2.0.tar.gz`
- `cd redis-3.2.0`
- `make`

Lancement

- `cd src`
- `./redis-server`

Connection

- `cd redis-3.2.0/src`
- `./redis-cli`

www.cnrs.fr

Fondamentaux

Clés

Valeurs

Clés

- Ne doit pas contenir des caractères
  - **Espace**
  - **Retour à la ligne**
- Conseils
  - **Utiliser « . » ou « - » à la place des caractères interdits**
  - **Pas de clé trop longue**
  - **Pas de clé trop courte**
  - **Format conseillé**
    - **« object-type:id »**

      Exemple : « **user:1000** »
    - **« object-type:id:field » pour simuler des champs**

      Exemple : « **user:1001:name** »

Valeurs

- String (chaînes de caractères)
- Atomic counter (compteurs numériques)

  Les données numériques sont stockées dans la base sous la forme de chaîne de caractères
- List (listes)
- Set (tableaux)
- Sorted set (tableaux ordonnés)
- Hash (dictionnaires)

www.cnrs.fr

Travaux Dirigés

1. Premiers pas
2. Stocker des données volatiles

www.cnrs.fr

Travaux Dirigés

2. Premiers pas

Clés

- Principales commandes
  - keys
  - exists
  - del (delete en python)
  - expire, expireat
  - object
  - persist
  - randomkey
  - rename
  - type
  - ttl
  - move

Valeurs String

- Valeur de tout type (JSON, valeur d'une image, entier)

  Cas d'utilisation des incréments

  - Compteur/séquence rapide
  - Vote
  - Génération d'identifiant
- Limité à 1Go

- Principales commandes
  - set, get, getset
  - mset, mget (set et get pour plusieurs clés)
  - incr, decr

Exercices

- 2.1 Connexion à la base de données
- 2.2 Insertion, recherche et suppression d'éléments String
  - Ex1

cles	valeur

  - Ex 2

clesnum	12

- 2.3 Incréments

  Pour simuler une table qui comptabilise les visites, préfixer les clés par "visites:"
  - Ex7
  - Ex8

visites:site	0
visites:pages:accueil	0
visites:pages:rbdd	0
visites:pages:devlog	0

www.cnrs.fr

Travaux Dirigés

3. Stocker des données volatiles

Données volatiles

- **Cas d'utilisation**
  - Stocker les clés de session Web pour gérer les durées de session
  - Stocker les paramètres et/ou résultats de recherche applicatives lancées par les utilisateurs

Exercices

- Objectif : Créer un cache pour les sessions utilisateurs

- Définir un délai d'expiration sur des clés

user1:session	valtemp5
user2:session	valtemp10
user3:session	Valtemp15

www.cnrs.fr

Travaux Dirigés

Twitter avec Redis

www.cnrs.fr

Travaux Dirigés

Twitter avec Redis

- Modélisation

Modélisation

Quels sont les données à stocker ?

- Réfléchir aux cas d'utilisation
- Modéliser les objets à manipuler

Modélisation

Quels sont les données à stocker ?

- Utilisateurs (users)
- Tweets
- Followers & amis
- Chronologie (timeline)
- Tendances (trends)

www.cnrs.fr

Travaux Dirigés

Twitter avec Redis

4. Dictionnaires (*Hash*)

Dictionnaires

- Un enregistrement contient plusieurs couples de clé-valeur
- Cas d'utilisation
  - Réponse aux besoins d'une base document simple

- Principales commandes (commencent par *H*)
  - hset, hget, hlen
  - hgetall (pour obtenir tous les couples clé-valeur)
  - hkeys et hvals (pour obtenir toutes les clefs ou valeurs)
  - hincrby (pour incrémenter un compteur)
  - hdel (pour faire le ménage)

Exercices

- Objectif : Gérer des utilisateurs

  Exercices 1 à 3

username	*firstname*	*lastname*	*country*
redistrainer	Guillaume	HARRY	France
mongotrainer	Olivier	PORTE	France
daboss	…	…	…

Exercices

- Objectif : Gérer des messages

  Exercice 4, 5

clé	*user_id*	*time*	*body*
post:001	daboss	16/05/2017	En pleine *#concentration*
post:002	redistrainer	17/05/2017	Formation *#nosql* #devlog #rbdd
post:032	daboss	17/05/2017	Vive les réseaux *#devlog #rbdd*
Post:064	redistrainer	17/05/2017	Et c'est pas fini !
Post:128	daboss	17/05/2017	*#nosql #migraine*

www.cnrs.fr

Travaux Dirigés

Twitter avec Redis

- Listes

Listes

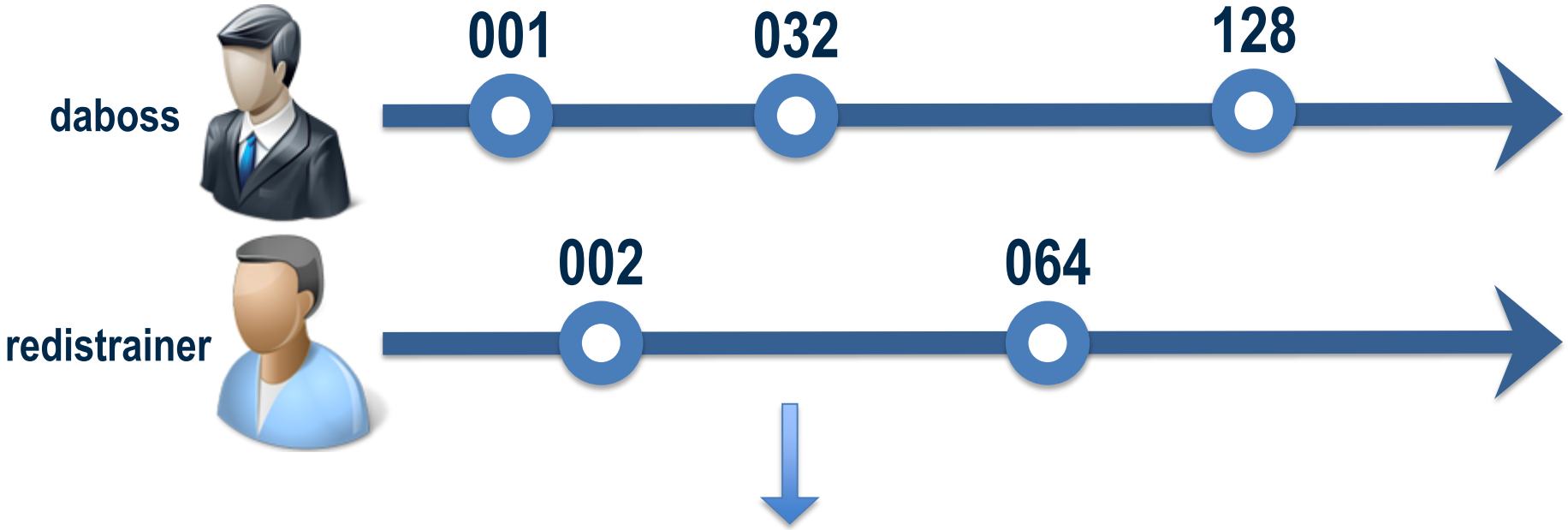
- Listes de chaînes de caractères
- Insertion/suppression possible au début et à la fin
- Accès à un élément de la liste par son index
- Empreinte mémoire optimisée pour les petites listes
- Cas d'utilisation
  - Sérialisation rapide des listes dans les applications
  - Indexation de listes de valeurs
  - Accélération des auto-complétion

Listes

- Principales commandes (commencent par *L*)
  - lpush, rpush (permettent respectivement d'ajouter un élément en début ou en fin de liste)
  - lrange (pour obtenir une partie des éléments de la liste)
  - lindex (pour obtenir un seul élément de la liste)
  - llen (pour obtenir la taille de la liste)

Exercices

- Objectif : gérer la chronologie des messages des utilisateurs



timeline:daboss	[128, 032, 001]
timeline:redistrainer	[064, 002]

www.cnrs.fr

Travaux Dirigés

Twitter avec Redis

6. Ensembles (*Set*)

Ensembles

- Collection d'objets non ordonnées
- Cas d'utilisation
  - Filtrer des ensembles entre eux
- Principales commandes (commencent par *S*)
  - sadd (pour ajouter une valeur)
  - scard (pour obtenir la taille (cardinalité))
  - smembers (pour lister les valeurs)
  - sinter (pour obtenir l'intersection entre 2 sets)
    - sinterstore (pour stocker dans un nouveau set l'intersection de 2 autres)
  - sunion (pour obtenir l'union entre 2 sets)
    - sunionstore (pour stocker dans un nouveau set l'union de 2 autres sets)
  - sdiff (pour obtenir la différence entre 2 sets)
    - sdiffstore (pour stocker dans un nouveau set la différence entre 2 sets)

Exercices

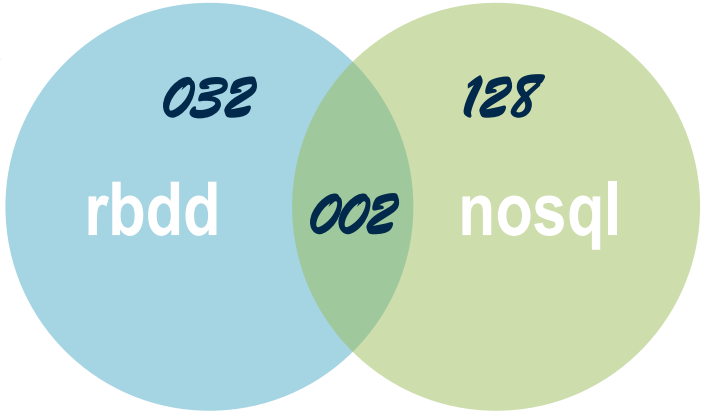
- Objectif : Gérer les mots clés

Ex1

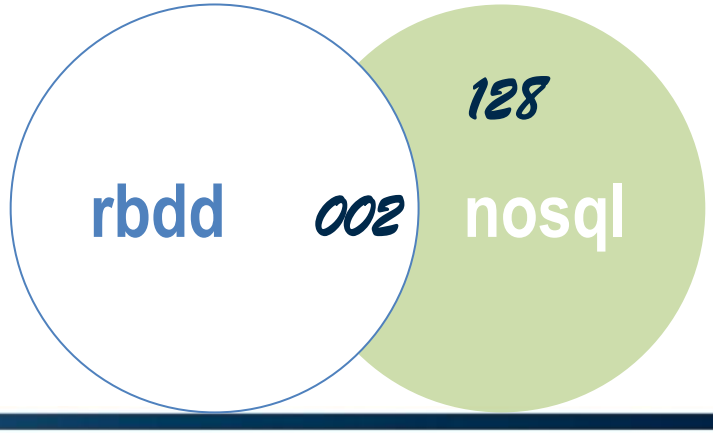
keyword:rbdd:messages	{ 002, 032 }
keyword:nosql:messages	{ 002, 128 }

Exercices

- Ex2 : Messages avec en commun les mots clés « rbbd » et « nosql »



- Ex3 : Si j'ai utilisé le mot clé « rbdd », les messages avec le mot clé « nosql » peuvent m'intéresser



www.cnrs.fr

Travaux Dirigés

Twitter avec Redis

- Ensembles ordonnés (*Sorted set*)

Ensembles ordonnés

- Collection d'objets ordonnées avec pondération
- Cas d'utilisation
  - Trier des ensembles par score
- Principales commandes (commencent par Z)
  - zadd, zcard, zinter, zunion
  - zrange
  - Zrangebyscore
  - zscore
  - zrank

Ensembles ordonnés

- Objectif : trouver les mots clés les plus populaires

keywords:score:11/01/2017	{ python *(15)*, nosql *(44)*, migraine *(50)*, 2017 *(23)* }
keywords:score:10/01/2017	{ nosql *(150)*, partitionnement *(17)*, disponibilité *(8)* }

- Trouver les 2 mots clés les plus utilisés aujourd'hui

  ['nosql', migraine']

- Trouver les mots clés commun des 2 jours triés par score avec la somme des scores

devloglrbdd	{nosql *(194)* }

www.cnrs.fr

Travaux Dirigés

C’est à vous

Exercices

- Créer la fonction de création de message
- Créer la fonction de création d'utilisateur
- Créer la fonction d'affichage d'un message
  - Format : « *username* a écrit le *time* : *body* »
- Créer la fonction de suivi de compte
  - Utiliser
    - Ensemble de qui l'utilisateur suit (friends)
    - Ensemble de par qui l'utilisateur est suivi (followers)
- Créer la fonction de proposition de compte à suivre

www.cnrs.fr

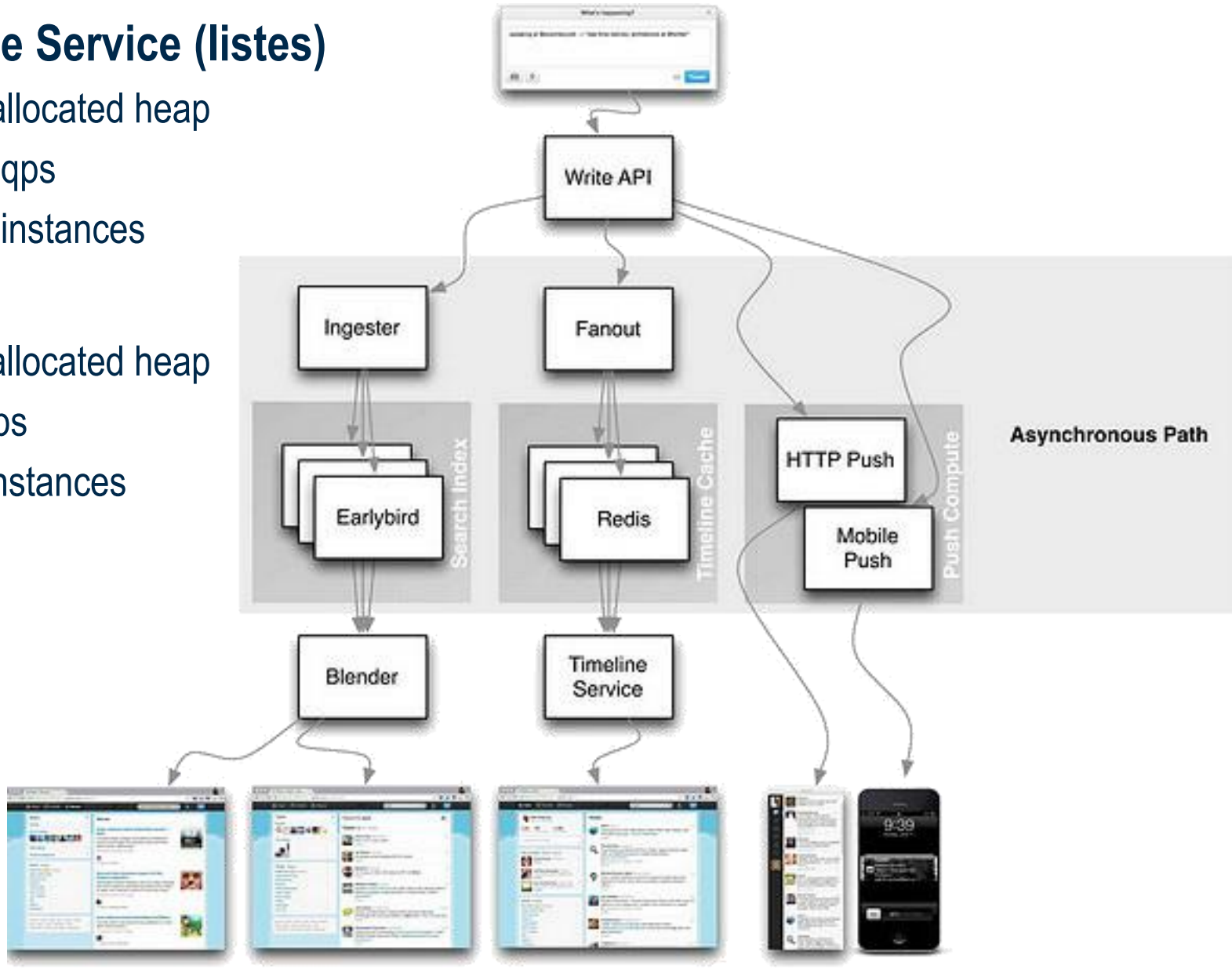
Conclusion

Conclusion

- Ajouter Redis dans votre architecture applicative pour
  - Gérer les données temporaires
  - Indexer
- Attention
  - L'intelligence est entièrement dans l'application
  - Pas de sharding natif
    - Librairies tierces
      - Ex : redis-shard

Exemple d'architecture : Twitter

- **Timeline Service (listes)**
  - ~40TB allocated heap
  - ~30MM qps
  - > 6,000 instances
- **BTree**
  - ~65TB allocated heap
  - ~9MM qps
  - >4,000 instances



Choisir la bonne structure

- **Strings**
  - Attention aux valeurs de grande taille qui peuvent subir des erreurs sur certains réseaux ou clients Redis
- **Listes**
  - Attention aux accès directs dans les listes de grande taille
- **Dictionnaires**
  - Attention à la gestion des clés secondaires. Il faut tenir à jour les métadonnées
- **Ensembles**
  - Attention à conserver une taille raisonnable
- **Ensembles ordonnés**
  - Ce type de structure est lourd en coût de traitement
- **Pubsub (pattern Publish/Subscribe)**
  - Moyen de broadcaster des messages