Projet AISE 2023

Le projet de cette année portera sur la réalisation d'un **stockage clef-valeur** performant en adoptant le même protocole que **Redis** (https://redis.io/docs/reference/protocol-spec/ (https://redis.io/docs/reference/protocol-spec/)).

Objectif

Le but de ce projet et de vous permettre d'expérimenter et de vous familiariser avec des concepts de programmation système et réseau dans une contrainte de fiabilité afin de proposer un service (ici, du stockage). Nous attendons que vous profitiez de cette occasion pour expérimenter et nous surprendre. Ainsi tout ajout dans la continuité du sujet que vous jugerez utile sera évalué. Durant les cours (et surtout les TPs) vous aurez *tous* les éléments pour atteindre la moyenne.

A noter qu'il sera possible dans un premier temps d'implémenter un protocole plus simple si vous le jugez utile. De plus toutes les commandes de Redis (https://redis.io/commands/ (https://redis.io/commands/)) ne sont pas attendues. Nous vous demandons de vous concentrer dans un premier temps sur les commandes suivantes:

- `PING`
- `SET`
- `GET`
- `DEL`

Détails d'implémentation

Vous pourrez réaliser cette implémentation dans le langage de votre choix. Garder à l'esprit qu'un point particulier sera attaché aux performances de votre implémentation, qui sera notamment testée avec de nombreuses clés/valeur. Nous vous recommandons donc des langages compilés comme Rust, C ou ses dérivés. Si vous retenez un langage interprété comme Python, il vous faudra justifier pourquoi vous considérer que ce choix technique est avantageux.

Support Attendu

Afin de vous aiguiller dans la réalisation ce travail, voici une liste des élements de notation prises en compte lors de la démonstration de votre projet en fin de module. Plus vous implémentez de points, et plus votre note sera haute:

Base

- Déploiement d'un serveur en écoute et gestion des requêtes client
- Support de plusieurs clients en simultané
- Support de la commande `PING`
- Support des commandes `SET` et `GET`
- Support de la commande `DEL`
- Système de tockage "en mémoire" performant

Intermédiaire

- Stockage de stockage fiable des clefs sur le disque (=persistance)
- Ajout de commandes et structures supplémentaires (toujours basé sur Redis)
- Types plus complexes (string, binaire)
- Support de l'atomicité dans le cas de clients multiples (requêtes concurrentes)

Avancé

- Support de la fédération de serveurs hachage consistant (https://en.wikipedia.org/wiki/Consistent_hashing)
- Mesure en contexte multi-clients distribué (N vers M)
- Mise en cache avancé des données (duplication)
- Gestion dynamique des clefs en mémoire et sur disque fonctionnement out-of-core (https://en.wikipedia.org/wiki/External_memory_algorithm)

Validation

Nous attendons une démonstration **pratique** de la base de données, le code sera envoyé en amont avant la présentation. Enfin, vous pourez comparer les performances de votre implémentation avec REDIS, si vous avez suffisament avancé, avec par exemple `redis-benchmark`. À noter que nous acceptons également un protocole plus simple, à condition de le justifier.

Rendu

- Modalités:
  - Par groupe de 2 ou 3 (limitez le nombre de groupe à 1)
  - Rendu via Forge Git (=dépôt privé Github par exemple)
  - Possibilité (mais non obligatoire) de fournir un rapport expliquant vos choix.
  - Envoyez-nous un email pour accéder à vos dépots:
    - **OBJET:** [AISE] Projet : [NOM 1] [NOM 2]...
    - jbbesnard@paratools.fr (mailto:jbbesnard@paratools.fr)
    - adamj@paratools.fr (mailto:adamj@paratools.fr)
  - **Pas.** De. **Zip.** Par. **Mail !**
- **Rendu**: 18 Décembre 2023 23h59
- **Démos**: 22 Décember 2023

Bon Courage !