Pourquoi Redis est rapide

Points Clés

- Il semble probable que la vidéo YouTube “System Design: Why is single-threaded Redis so fast?”de ByteByteGo explique la rapidité de Redis grâce au stockage en mémoire, au multiplexage d’E/S et aux structures de données efficaces.
- La recherche suggère que ces facteurs permettent à Redis de traiter jusqu’à 100 000 requêtes par seconde, malgré le fait qu’il soit monothreadé pour le traitement des requêtes.
- Les preuves penchent en faveur des versions récentes de Redis ajoutant du multithreading pour des tâches spécifiques, mais le cœur reste monothreadé.

Introduction

Cet article de blog est basé sur la vidéo YouTube “System Design: Why is single-threaded Redis so fast?” de ByteByteGo, faisant partie de leur série sur la conception de systèmes. Redis, connu pour ses hautes performances, peut traiter jusqu’à 100 000 requêtes par seconde sur une seule machine, ce qui est impressionnant pour un système monothreadé. Explorons pourquoi cela est possible et ce qui rend Redis si rapide.

Raisons de la Vitesse de Redis

La vitesse de Redis peut être attribuée à plusieurs facteurs clés, probablement couverts dans la vidéo :

- **Stockage en Mémoire** : Redis stocke les données en RAM, qui est beaucoup plus rapide que le stockage sur disque. Cela réduit la latence et augmente le débit, car les temps d’accès à la mémoire sont de l’ordre des nanosecondes, contre des millisecondes pour l’accès au disque.
- **Multiplexage d’E/S et Exécution Monothreadée** : Le multiplexage d’E/S, utilisant des mécanismes comme epoll sur Linux, permet à un seul thread de gérer plusieurs connexions client de manière efficace. Cela évite le surcoût de la commutation de contexte, et la boucle monothreadée simplifie les opérations en éliminant les problèmes de synchronisation.
- **Structures de Données Efficaces** : Redis utilise des structures de données optimisées comme les tables de hachage (recherches O(1)), les listes chaînées et les listes de saut, qui améliorent les performances en minimisant l’utilisation de la mémoire et en accélérant les opérations.

Évolution et Mise à l'Échelle

Pour une haute concurrence, Redis peut être mis à l'échelle horizontalement en utilisant plusieurs instances ou en utilisant un clustering. Un détail inattendu est que bien que le traitement des requêtes reste monothreadé, les versions récentes (depuis 4.0) ont introduit le multithreading pour des tâches comme la suppression d'objets en arrière-plan, améliorant encore les performances sans changer le modèle principal.

Note de Sondage : Analyse Détaillée de la Performance Monothreadée de Redis

Cette section fournit une analyse complète de pourquoi Redis monothreadé est si rapide, basée sur la vidéo YouTube "System Design: Why is single-threaded Redis so fast?"de ByteByteGo et des recherches connexes. La vidéo, publiée le 13 août 2022, fait partie d'une série axée sur la conception de systèmes, rédigée par les créateurs des livres à succès sur les entretiens de conception de systèmes. Étant donné la focalisation du canal, la vidéo fournit probablement des informations détaillées adaptées aux entretiens techniques et aux discussions sur la conception de systèmes.

Contexte et Contexte Redis, un magasin de clés-valeurs en mémoire open-source, est largement utilisé comme cache, courtier de messages et moteur de streaming. Il prend en charge des structures de données comme les chaînes, les listes, les ensembles, les hachages, les ensembles triés et des structures probabilistes comme le filtre de Bloom et HyperLogLog. Le titre de la vidéo suggère une exploration de pourquoi Redis maintient des performances élevées malgré son traitement de requêtes monothreadé, qui est central à sa conception.

D'après des articles connexes, Redis peut traiter jusqu'à 100 000 requêtes par seconde (QPS) sur une seule machine, une figure souvent citée dans les benchmarks de performance. Cette vitesse est surprenante étant donné le modèle monothreadé, mais des recherches indiquent que cela est dû à plusieurs choix architecturaux.

Facteurs Clés Contribuant à la Vitesse de Redis

1. **Stockage en Mémoire** Redis stocke les données en RAM, qui est au moins 1000 fois plus rapide que l'accès aléatoire au disque. Cela élimine la latence de l'E/S disque, avec des temps d'accès à la RAM autour de 100-120 nanosecondes contre 50-150 microsecondes pour les SSD et 1-10 millisecondes pour les HDD. La vidéo met probablement l'accent sur cela comme une raison principale, car cela s'aligne avec la focalisation du canal sur les fondamentaux de la conception de systèmes.

Aspect	Détails
Support de stockage	RAM (en mémoire)

Aspect	Détails
Temps d'accès	~100-120 nanosecondes
Comparaison avec le disque	1000x plus rapide que l'accès aléatoire au disque
Impact sur les performances	Réduit la latence, augmente le débit

2. **Boucle d'Exécution Monothreadée et Multiplexage d'E/S** Le multiplexage d'E/S permet à un seul thread de surveiller plusieurs flux d'E/S simultanément à l'aide d'appels système comme `select`, `poll`, `epoll` (Linux), `kqueue` (Mac OS) ou `evport` (Solaris). Cela est crucial pour gérer plusieurs connexions client sans blocage, un point probablement détaillé dans la vidéo. La boucle d'exécution monothreadée évite le surcoût de la commutation de contexte et de la synchronisation, simplifiant ainsi le développement et le débogage.

Mécanisme	Description
epoll/kqueue	Efficace pour une haute concurrence, non bloquant
select/poll	Plus ancien, moins évolutif, complexité O(n)
Impact	Réduit le surcoût de connexion, permet le pipeline

Cependant, les commandes bloquant le client comme `BLPOP` ou `BRPOP` peuvent retarder le trafic, un inconvénient potentiel mentionné dans des articles connexes. La vidéo peut discuter de la manière dont ce choix de conception équilibre la simplicité avec les performances.

3. **Structures de Données de Bas Niveau Efficaces** Redis utilise des structures de données comme les tables de hachage pour des recherches de clés O(1), les listes chaînées pour les listes et les listes de saut pour les ensembles triés. Elles sont optimisées pour les opérations en mémoire, minimisant l'utilisation de la mémoire et maximisant la vitesse. La vidéo inclut probablement des diagrammes ou des exemples, comme la manière dont les tables de hachage permettent des opérations clé-valeur rapides, un sujet courant dans les entretiens de conception de systèmes.

Structure de Données	Cas d'utilisation	Complexité de Temps
Table de Hachage	Stockage clé-valeur	O(1) moyenne
Liste Chaînée	Listes, efficace aux extrémités	O(1) pour les extrémités
Liste de Saut	Ensembles triés, stockage ordonné	O(log n)

Cette optimisation est cruciale, car la plupart des opérations Redis sont basées sur la mémoire, avec les goulots d'étranglement se situant généralement dans la mémoire ou le réseau, et non dans le CPU.

Considérations Supplémentaires et Évolution Bien que le traitement des requêtes soit monothreadé, les versions récentes de Redis ont introduit le multithreading pour des tâches spécifiques. Depuis Redis 4.0, la libération asynchrone de la mémoire (lazy-free) a été mise en œuvre, et depuis 6.0, le multithreading pour l'analyse du protocole sous haute concurrence a été ajouté. Ces changements, probablement mentionnés dans la vidéo, améliorent les performances sans altérer le modèle monothreadé pour les opérations principales.

Pour une mise à l'échelle au-delà d'une seule instance, Redis prend en charge le clustering et l'exécution de plusieurs instances, une stratégie qui peut être discutée pour répondre aux besoins de haute concurrence. C'est un aspect important pour la conception de systèmes, s'alignant avec la focalisation du canal sur les systèmes à grande échelle.

Inconvénients Potentiels et Comparaisons Le modèle monothreadé a des avantages comme l'absence de contention de verrouillage et un débogage plus simple, mais il peut rencontrer des défis avec les opérations bloquantes et les goulots d'étranglement mémoire/réseau sous charge élevée. Des articles connexes suggèrent que pour des tâches intensives en CPU, des bases de données multithreadées pourraient mieux performer, mais pour les cas d'utilisation typiques de Redis, la conception monothreadée est optimale.

Conclusion La vidéo "System Design: Why is single-threaded Redis so fast?"de ByteByteGo couvre probablement le stockage en mémoire, le multiplexage d'E/S et les structures de données efficaces comme raisons clés de la rapidité de Redis. Ces facteurs lui permettent de gérer un haut QPS, avec des versions récentes ajoutant du multithreading pour des optimisations spécifiques. Cette analyse fournit une compréhension complète, adaptée à la fois aux apprenants techniques et aux professionnels de la conception de systèmes.

Citations Clés

- Pourquoi Redis est-il si rapide blog post
- Pourquoi Redis est-il si rapide malgré son caractère monothreadé article
- Entretien sur le modèle de thread de Redis article
- Pourquoi Redis monothreadé est-il si rapide article