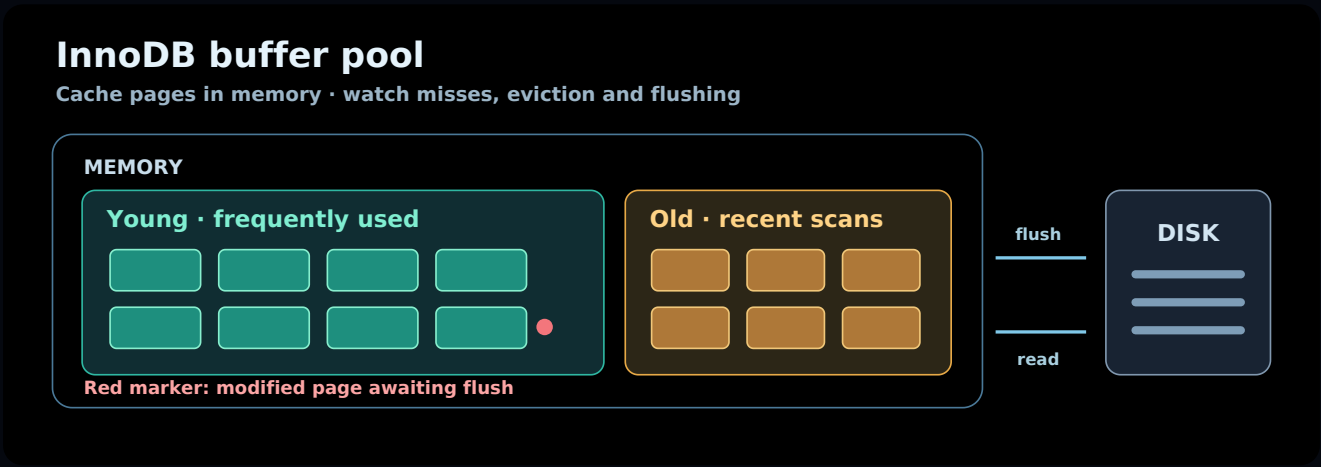
Comprendre le buffer pool InnoDB : guide d'optimisation des performances

Duthaho · 28 septembre 2026

MYSQL INNODB BUFFER-POOL PERFORMANCE



InnoDB s'appuie sur la mémoire pour éviter de relire chaque page sur disque. Son buffer pool conserve les pages de données et d'index récemment utilisées ; les écritures modifient ces pages en mémoire avant leur vidage sur disque. Comprendre quelles pages entrent, lesquelles sortent et combien de temps le cache met à se réchauffer aide à expliquer les performances en régime normal comme les lenteurs après redémarrage.

Ce que contient le buffer pool

`innodb_buffer_pool_size` fixe la taille de cette zone de mémoire partagée. Les pages InnoDB font généralement 16 Kio, mais `innodb_page_size` peut être configuré différemment lors de la création de l'instance. Les pages de données et d'index partagent l'espace avec des structures internes. Une page déjà en mémoire évite une lecture du fichier de données ; une page absente doit être chargée depuis le stockage.

Sur un serveur dédié à la base de données, un grand pool peut être utile, mais aucun pourcentage fixe de la RAM ne convient partout. Il faut laisser de la mémoire au système, aux connexions, aux opérations temporaires, aux autres moteurs et aux services voisins. Un pool de 48 Gio sur un serveur de 64 Gio est **un exemple à tester**, pas une recommandation universelle.

Mesurez les lectures sur un même intervalle, plutôt que de vous fier à un pourcentage calculé depuis le démarrage :

```
SHOW GLOBAL STATUS WHERE Variable_name IN (
  'Innodb_buffer_pool_read_requests',
  'Innodb_buffer_pool_reads',
  'Innodb_buffer_pool_pages_dirty',
  'Innodb_buffer_pool_wait_free'
);
```

Le taux de défaut de cache sur l'intervalle vaut `ΔInnodb_buffer_pool_reads / ΔInnodb_buffer_pool_read_requests`. Si ce taux augmente avec les lectures physiques et la latence des requêtes, l'ensemble des données actives ne tient peut-être plus correctement en mémoire. Un taux de succès élevé ne prouve pas, à lui seul, que les requêtes sont rapides. Les pages sales et `Innodb_buffer_pool_wait_free` éclairent la pression de vidage.

Pourquoi un grand scan ne chasse pas forcément toutes les pages actives

InnoDB utilise une liste de type LRU modifiée. Les nouvelles pages arrivent dans la partie **old** ; les pages régulièrement consultées peuvent monter dans la partie **young**. Cette insertion au milieu protège une partie du jeu de travail actif contre un scan ponctuel.

`innodb_old_blocks_pct` règle la proportion réservée à la partie old (37 par défaut dans MySQL 8.4). `innodb_old_blocks_time` impose un délai avant la promotion d'une nouvelle page (1000 ms par défaut dans MySQL 8.4). Un délai plus long peut protéger les pages OLTP pendant un gros rapport, mais aussi retarder la promotion de pages réellement utiles. Le mettre à zéro supprime ce délai ; ce n'est pas une solution générale aux charges de lecture. Mesurez la charge réelle avant de modifier ces paramètres.

Réchauffer le pool après redémarrage ou bascule

Après un redémarrage, le cache est froid. Les premières requêtes peuvent lire beaucoup de fichiers de données, même si le SQL et les index n'ont pas changé. InnoDB peut enregistrer les **identifiants** des pages à l'arrêt et les recharger au démarrage avec `innodb_buffer_pool_dump_at_shutdown` et `innodb_buffer_pool_load_at_startup`.

`innodb_buffer_pool_dump_pct` détermine la proportion des pages les plus actives à enregistrer. Le dump ne contient pas les pages elles-mêmes : le rechargement effectue toujours des lectures disque.

On peut aussi laisser le trafic normal réchauffer le cache, lancer des requêtes représentatives bien bornées ou préparer un réplica avant d'y déplacer le trafic. Un préchargement peut concurrencer la production sur les E/S : limitez son débit et surveillez la latence. Un pool rempli à 80 % n'est pas nécessairement chaud pour les requêtes importantes. Jugez l'état de préparation avec la latence et les lectures physiques de requêtes représentatives, ainsi qu'avec l'état du chargement du pool si vous l'utilisez.

Écritures, redo et autres structures

Lorsqu'une transaction modifie une page, sa copie en cache devient **sale**. InnoDB écrit des enregistrements redo afin de rejouer les changements validés après une panne ; le vidage en arrière-plan écrit ensuite les pages modifiées dans les fichiers de données. Ce vidage fait avancer le checkpoint. Le redo log et le buffer pool ont donc des rôles différents : une cible trop agressive pour les pages sales augmente les E/S, alors qu'un vidage insuffisant peut créer de la pression plus tard. Conservez une politique de durabilité adaptée ; `innodb_flush_log_at_trx_commit=1` est le choix habituel lorsque chaque transaction validée doit survivre à une panne du serveur.

Le **change buffer** peut différer les modifications de pages d'index secondaires non uniques absentes de la mémoire. Il peut réduire les E/S aléatoires de certaines charges d'écriture MySQL, au prix de fusions ultérieures et d'une consommation mémoire. L'**adaptive hash index** peut accélérer des recherches répétées, mais consomme aussi de la mémoire et peut entraîner de la contention. Aucun des deux n'est bénéfique par défaut : vérifiez sa disponibilité selon la version, observez les compteurs et testez avec un trafic représentatif. En particulier, MariaDB 10.5 et les versions suivantes utilisent une seule instance de buffer pool : le conseil de régler `innodb_buffer_pool_instances=8` ne s'y applique pas.

Une méthode de réglage concrète

Pour un système de commandes très sollicité, partez du véritable jeu de travail et d'un budget mémoire. Relevez la latence, les lectures physiques, les pages sales et les E/S en charge normale et en pointe. Si les lectures disque montent quand le pool est plein, améliorez d'abord les index et les accès SQL, puis testez un pool plus grand en surveillant la mémoire totale du serveur. Si un

scan analytique chasse les pages actives, testez les paramètres de la partie old. Si la latence ne monte qu'après un redémarrage ou une promotion, testez dump/load et réchauffez le futur primaire avant d'y router la production.

Dans MySQL 8.4, plusieurs instances du pool peuvent réduire la contention sur les grands pools, mais leur nombre doit répondre à des mesures et laisser environ 1 Gio ou plus par instance. Une cible globale de succès du cache telle que 99 % est un indice de diagnostic, jamais un objectif de service à elle seule. Comparez chaque changement sur la même charge en incluant les gains de vitesse et les régressions de mémoire ou d'E/S.

Conclusion

Le buffer pool est plus qu'un grand cache. Sa politique d'admission protège les pages souvent utilisées, son réchauffement détermine la latence après redémarrage et son vidage relie les écritures en mémoire au stockage durable. Réglez-le à partir de mesures de votre charge et avec une marge mémoire suffisante, plutôt qu'en suivant une recette universelle.

Adapté et traduit avec l'accord de l'auteur à partir de l'article de Duthaho sur Medium, publié initialement le 22 août 2025. Les détails propres aux versions ont été vérifiés dans la documentation MySQL 8.4 sur le buffer pool et la documentation MariaDB sur le buffer pool.