Configuration par défaut

Historiquement, la configuration initiale de PostgreSQL était dimensionnée pour supporter les plus vieilles variantes d'UNIX où l'allocation de grande quantité de mémoire n'était pas possible. Sur les systèmes modernes qui possèdent beaucoup de mémoire libre, cela handicap fortement les performances d'un PostgreSQL non configuré.

Les valeurs par défaut sont beaucoup moins pessimistes dans les versions récentes. Désormais la configuration système est examinée au moment de l'initialisation de la base de la données et plus de mémoire sera allouée s'il est possible de le faire. De plus, des récents changements dans la gestion du cache permettent même une utilisation plus efficace du cache en quantité modeste.

Sur les deux systèmes un réglage essentiel concerne l'allocation d'un bloc de mémoire partagée dédiée au SGBD. MySQL règle cela avec key_buffer_size en utilisant MyISAM, et innodb_buffer_pool_size avec InnoDB. PostgreSQL taille son espace mémoire principal avec shared_buffers. Sur un serveur de type linux de génération actuelle, il est admis qu'une version récente de PostgreSQL assigne au moins 24 Mo par défaut à shared_buffers lors de la création du cluster.

Pour un serveur moderne dédié, le principe de base pour PostgreSQL et MySQL est de dimensionner la mémoire dédiée à au moins 1/4 de la mémoire totale de la machine, pouvant grimper à 1/2 de la RAM pour des quantités supérieures à la normale. Il n'est cependant pas hors de question de pousser ce pourcentage encore plus haut sur des systèmes avec une quantité vraiment grande de RAM; Le guide de MySQL InnoDB suggère même que 80% n'est pas déraisonnable.

Tests de performances

Sun Microsystems a récemment publié les résultats de ses tests en utilisant PostgreSQL et MySQL. Le peu de différences matériels entre les deux systèmes suffit pour ne pas comparer les deux résultats directement. Mais le fait que les deux résultats soient assez proches avec une configuration similaire indique que, malgré des performances différentes entre les deux bases, l'importance de cette différence n'est pas particulièrement grande avec ce type d'application.

Par comparaison, un Oracle sur HP offre des résultats comparable en performance sur du matériel moins impressionnant, suggérant ainsi que les deux bases de données open-source ont toujours du retard sur le meilleur des produits commerciaux en terme d'efficacité des performances.

Lorque l'on fait une comparaison équitable en incluant le coût des licences, la "performance par euro" paraît semblable pour PostgreSQL et MySQL et très supérieures à la moyenne dans l'industrie de base de données.

Ce document est basé sur un texte éponyme écrit par Greg Smith, Christopher Browne, Lukas Kahwe Smith, ainsi que d'autres membres de la communauté PostgreSQL.

http://www.postgresql.org/docs/techdocs.83

Cet article a été traduit en français par l'association PostgreSQLFr (notamment Guillaume de Rorthais et Christophe Chauvet)

http://www.postgresqlfr.org/?q=node/1432

Pourquoi préférer PostgreSQL à MySQL

Un comparatif

de fiablité et de rapidité

en 2007

http://www.postgreSQLfr.org

Introduction

Pendant des années, le marché considérait MySQL comme plus rapide et plus facile à utiliser que PostgreSQL. PostgreSQL était perçu comme étant plus puissant, focalisé sur l'intégrité des données, et plus respectueux des normes SQL, mais également plus lent et plus compliqué à utiliser.
Ces perceptions appartiennent au passé, et avec les versions actuelles, les choses ne sont plus aussi tranchées qu'auparavant.

• MySQL 5.0 (Oct 2005) a finalement ajouté un « strict mode » pour réduire l'écart en terme d'intégrité des données et de conformité aux spécifications. Le support des procédures stockées, vues, déclencheurs et curseurs,
• PostgreSQL 8.1 (Nov 2005) apporte d'importantes améliorations de performance, particulièrement en terme d'évolutivité. L'attention a principalement été portée sur l'amélioration des performances pour toutes les versions 8.X et ce jusqu'à l'actuelle 8.2.

Alors que l'innovation sur ces deux SGBD a progressé, chacune des communautés de développement a travaillé activement à réduire la liste de ses désavantages perçus. Ce document vise à clarifier les situations dans lesquelles PostgreSQL est plus approprié que MySQL, en essayant de comparer équitablement les versions de production actuelles et d'en discuter les forces et faiblesses.

La position de ce document est que lorsque nous comparons les deux pour une application sérieuse exigeant un haut degré d'intégrité des données dans une base transactionnelle, la génération actuelle de PostgreSQL s'exécutera aussi bien ou mieux que MySQL, tout en conservant son avance dans sa conformité aux standards SQL et sa richesse de fonctionnalité. Il est aussi espéré qu'en explorant les différences entre les deux systèmes, vous arriviez à apprécier l'approche fondamentale de l'équipe de conception de PostgreSQL qui s'évertue à prioriser un comportement fiable et prévisible.

Comparaison des versions, ensemble de fonctionnalités et détails

Les versions actuelles recommandées en production sont PostgreSQL 8.2 et MySQL 5.0, et feront donc ici l'objet de notre comparaison. PostgreSQL 8.1 et 8.2 sont aujourd'hui les deux versions supportées avec de bonnes performances : 8.2 est sensiblement plus rapide, mais déployer 8.1 reste une option tout à fait viable pour le moment, particulièrement dû au fait que les vendeurs de système d'exploitation la distribuent et la prennent en charge plus souvent que la 8.2 encore relativement récente.

Les fonctionnalités des deux produits dans des domaines non fondamentaux ne seront pas abordées ici. Même si les listes des fonctionnalités sont utiles, certains comportements internes nécessitent une compréhension profonde des systèmes respectifs pour bien les saisir.

Un autre sujet qui sort du cadre de ce document est qu'un nombre plus important d'applications choisissent MySQL comme base de données plutôt que PostgreSQL, et ceci est certainement un facteur d'influence important sur le choix de la base la plus adaptée à une situation particulière.

Fiabilité

Intégrité des données

Avant la version 5.0, MySQL méritait clairement sa réputation à propos de l'incohérence des données insérées dans la base. La version actuelle dispose d'un mode SQL strict et tout client MySQL a la possibilité de changer son mode SQL et contourner ainsi ce comportement, avec comme conséquences que ces validations de contraintes ne soient plus forcément assurées coté serveur.

PostgreSQL a toujours été strict sur la validation des données avant de les insérer dans la base de données, et il n'existe aucune alternative au client pour contourner ces vérifications.

Clés étrangères

L'implémentation correcte des techniques de conception comme les formes normales repose sur la capacité de la base de données à utiliser les clés étrangères pour représenter les relations entre les tables.

Avec MySQL, seul InnoDB supporte les clés étrangères. Un problème avec leur implémentation est qu'elle est limitée et ignorera silencieusement plusieurs syntaxes standard. La philosophie de conception à la base de PostgreSQL est de produire des erreurs ou des avertissements dans les situations similaires où une opération est ambigüe ou non supportée.

Transactions

MyISAM est le composant de MySQL à l'origine de sa réputation de rapidité. Ce moteur a d'excellentes performances en lecture et son analyseur est très efficace pour les requêtes simples, ce qui le rend très rapide pour les applications en lecture intensive. Cependant, il est communément connu que MyISAM est plus vulnérable aux corruptions de données que la plupart des bases de données sérieuses, et qu'en cas d'incident, le temps de redémarrage est non négligeable.

L'intégration du moteur de stockage InnoDB à MySQL a grandement surpassé MyISAM en terme d'intégrité des données, ajoutant un mécanisme de ré-exécution des journaux plus robuste et le support des transactions ACID. Cependant, cette nouvelle approche apporte aussi beaucoup plus de charge, et les tables InnoDB ne sont pas aussi rapides que les MyISAM. De plus, les tables des métadonnées internes à MySQL sont toujours stockées en MyISAM.

PostgreSQL a toujours porté attention à l'intégrité des données au niveau transactionnel, se gardant ainsi des problèmes de verrou au minimum, et empêchant une erreur matériel ou une configuration extrêmement mauvaise de corrompre les données. Enfin PostgreSQL intègre entièrement son moteur de base de donnée, alors que InnoDB est un produit sous licence double actuellement détenu par Oracle.

Fonctionnalités

DDL transactionnel

Avec PostgreSQL, lorsque vous êtes à l'intérieur d'une transaction presque toute opération peut être annulée. Il existe quelque opérations irréversibles, mais les modifications classiques de table peuvent être défaites en exécutant un ROLLBACK. Cela s'applique aussi aux importantes modifications DDL comme la création de tables.

MySQL ne supporte aucun type d'annulation en utilisant MyISAM. Avec InnoDB, le serveur déclenche une validation implicite même si le comportement normal d'auto-commit est désactivé.

Verrou de transaction et extensibilité

PostgreSQL utilise un modèle de verrous robuste appelé MVCC qui limite les situations où les clients interfèrent les uns avec les autres. Un court résumé du principal bénéfice du MVCC serait « les lecteurs ne sont jamais bloqués par les écritures ». Le niveau d'isolation des transactions par défaut est "read commited" (Lecture de données validées).

InnoDB de MySQL implémente MVCC en utilisant un espace d'annulation (rollback segment), inspiré par la conception d'Oracle ; leur nouveau moteur Falcon fonctionne de la même manière. Les bases de données InnoDB supportent les quatre standards d'isolation de transaction SQL, celui par défaut étant « Repeatable Read » (Lecture répétée).

Lorsque l'on compare les deux modèles, PostgreSQL assure une séparation des clients tel que les données traitées soient toujours cohérentes dans toutes les circonstances. MySQL autorise des configurations où le code d'un client qui ne valide pas correctement ses transactions peut aboutir à une vue des données qui serait considérée comme incohérente par les standards stricts de PostgreSQL. Cependant, dans les situations où il est acceptable d'avoir des lectures avec de petites incohérences, avoir la possibilité d'utiliser des verrous moins stricts peut être un avantage en terme de performances dans MySQL.

En parti à cause de l'implémentation très mature des verrous dans PostgreSQL, même dans les situations où MySQL paraît initialement plus rapide PostgreSQL peut aller plus loin et arriver à un débit plus élevé lorsque le nombre d'utilisateurs simultanés devient important.

Jointures complexes

PostgreSQL utilise une méthode économique d'optimisation des requêtes afin d'augmenter les performances pour les différents types de jointures. Le coût des requêtes est évalué à partir des statistiques du planificateur et des mécanismes avancés tels que le Genetic Query Optimizer.

Le planificateur de MySQL n'a pas ce niveau de sophistication, En utilisant des astuces sur les index on peut s'assurer que les jointures se fassent correctement. Pour faciliter cette tâche, MySQL fournit un Query Profiler qui typiquement facilite le travail sur ces données EXPLAIN. En dehors de ces astuces, l'optimisation des sous-selections est une faiblesse connue de MySQL.