Introduction

S. Lopes, complété par L.Yeh

January 28, 2021

Content

Qu'est ce que le Tuning ?

- Tuning = réglage
- Tuning de bases de données: faire en sorte que le système (les applications) soit **plus performant**.
- *plus performant*: différents critères possibles (dépendent des applications)
  - nombre de transactions par seconde (*throughput*)
  - temps de réponse (*response time*)
  - ...
- Pourquoi ? Pour que les utilisateurs soient satisfaits !

Qu'est ce que le Tuning ?

- Intervention à différents niveaux
  - Applications: la façon dont elles sont construites.
    Example 1 *Interaction avec l'utilisateur dans une transaction.*
  - Organisation des données.
    Example 2 *Définition d'un index, dénormalisation, partitionnement, ...*
  - Paramètres du système de gestion de bases de données (SGBD).
    Example 3 *Taille du cache de données.*
  - Configuration du système d'exploitation (SE).
    Example 4 *Buffer d'entrée/sortie au niveau du SE.*
  - Matériel.
    Example 5 *Taille et performances du ou des disques.*

  $\Rightarrow$ Nécessite une connaissance approfondie et vaste (transversale).

Références

1. *Database tuning: Principles, Experiments, and Troubleshooting Techniques*. Dennis Shasha, Philippe Bonnet. Morgan Kaufmann. 2002.
2. Manuels du SGBD: *Oracle Designing and Tuning for Performance*, ...
3. *Database Administration: the Complete Guide to Practice and Procedures*. Graig Mullins. Addison Wesley. 2002.
4. Manuels du SGBD: *Oracle Administrator's Guide*, ...

Objectifs du cours

- Acquisition des principes fondamentaux du tuning.
  - Impact des concepts déjà étudiés (transactions, optimisation de requêtes, dénormalisation, ...) sur les performances.
  - Comment collecter les informations utiles ?
  - Comment analyser ces informations ?
  - Comment intervenir pour corriger un problème ?
- Rester indépendant du SGBD afin d'être applicable dans tous les cas.
- Mettre en application sous Oracle pour avoir un exemple concret.

Plan du cours

1. Introduction (principes de base du tuning)
2. Notions d'administration de BD
3. Aspects systèmes
   - Matériel (disques, RAM, processeur)
   - SE
   - SGBD (paramètres, transactions (contrôle de concurrence, reprise))
4. Aspects physiques
   - Index
   - Organisation des données
   - Optimisation de requêtes
5. Aspects logiques
   - Dénormalisation
   - Partitionnement
   - Extraction de contraintes d'intégrité

- Le tuning repose sur le bon sens ⇒ à la fois facile et difficile.
  - **Facile**
    - pas de formules et de théorèmes compliqués.
    - formalisation trop poussée ⇒ hypothèses simplificatrices trop fortes.
    - quelques résultats tout de même.
  - **Difficile**
    - fait appel à des connaissances sur les applications, le SGBD, le SE et le matériel.

    ⇒ connaissances vastes et approfondies
- Important: il est nécessaire de comprendre le pourquoi et de ne pas appliquer une règle sans réfléchir.

Example 6 *Ne pas utiliser de fonctions d'agrégation*

- Le tuning repose sur le bon sens $\Rightarrow$ à la fois facile et difficile.
  - **Facile**
    - pas de formules et de théorèmes compliqués.
    - formalisation trop poussée $\Rightarrow$ hypothèses simplificatrices trop fortes.
    - quelques résultats tout de même.
  - **Difficile**
    - fait appel à des connaissances sur les applications, le SGBD, le SE et le matériel.

    $\Rightarrow$ connaissances vastes et approfondies
- Important: il est nécessaire de comprendre le pourquoi et de ne pas appliquer une règle sans réfléchir.

Example 6 *Ne pas utiliser de fonctions d'agrégation (provoque une baisse du temps de réponse) car un scan de beaucoup de tuples peut ralentir d'autres requêtes. Pas génant s'il y a peu de tuples ou un index.*

- Cinq principes de base.
  1. P1: Avoir une vision globale; intervenir localement
  2. P2: Partitionner élimine les goulots d'étranglement
  3. P3: Coûts d'initialisations élevés; coûts d'exécution sont bas
  4. P4: Faire sur le serveur ce qui doit être fait sur le serveur
  5. P5: Etre prêt à faire des compromis

P1: Avoir une vision globale; intervenir localement

- Identification précise du problème $\Rightarrow$ Mesurer les bonnes quantités.
- Intervention minimaliste $\Rightarrow$ Faire le bon diagnostic.

Example 7 *On observe les statistiques matérielles et on constate une forte activité disque. Une réaction simpliste serait d'acheter immédiatement plus de disques.*

P1: Avoir une vision globale; intervenir localement

- Identification précise du problème $\Rightarrow$ Mesurer les bonnes quantités.
- Intervention minimaliste $\Rightarrow$ Faire le bon diagnostic.

Example 7 *On observe les statistiques matérielles et on constate une forte activité disque. Une réaction simpliste serait d'acheter immédiatement plus de disques.*
Cependant, la forte activité peut être due à l'absence ou à la non utilisation d'un index. Dans ce cas, créer l'index ou forcer son utilisation est une solution plus économique et plus efficace.

Example 8 *On constate que le temps d'exécution d'une requête est élevé. On essaie donc de le réduire.*

P1: Avoir une vision globale; intervenir localement

- Identification précise du problème ⇒ Mesurer les bonnes quantités.
- Intervention minimaliste ⇒ Faire le bon diagnostic.

Example 7 *On observe les statistiques matérielles et on constate une forte activité disque. Une réaction simpliste serait d'acheter immédiatement plus de disques.*
Cependant, la forte activité peut être due à l'absence ou à la non utilisation d'un index. Dans ce cas, créer l'index ou forcer son utilisation est une solution plus économique et plus efficace.

Example 8 *On constate que le temps d'exécution d'une requête est élevé. On essaie donc de le réduire.*
Cependant, il faut s'assurer que cette requête est exécutée suffisamment (si la requête représente 1% du temps et que l'on divise par deux son temps d'exécution, le système ne sera amélioré que de 0,5%). On doit donc d'abord s'assurer que la requête est importante.

P2: Partitionner élimine les goulots d'étranglement

- Tous les composants d'un système sont rarement saturés en même temps
- Généralement, seule une petite partie du système est en cause
- Goulots d'étranglement: partie saturée d'un système (embouteillage sur la route)
- Solutions
  1. aller plus vite $\Leftrightarrow$ intervention locale (la première solution à essayer)
  2. créer plusieurs passages ou changer les horaires d'accès pour certains usagers $\Leftrightarrow$ partitionnement
- Partitionnement: répartition de la charge (plus de ressources, dans le temps, . . .)
- *Quand un goulot d'étranglement est localisé, essayer d'accélérer ses composants; si ca ne fonctionne pas, partitionner*

P2: Partitionner élimine les goulots d'étranglement

Example 9

Spatial : *un système pour chaque agence d'une banque.*
Logique : *chaque processus accède à une liste de blocs libres au hasard.*
Temporel : *une transaction longue s'exécute quand il y a moins de transactions courtes en cours d'exécution.*

P3: Coûts d'initialisations élevés; coûts d'exécution sont bas

- La phase de démarrage est souvent coûteuse
  - *Essayer d'obtenir les mêmes effets avec le moins de démarrages possibles*

Example 10

Disque : *éviter la fragmentation, partitionnement vertical.*
Réseau : *envoyer plusieurs petits messages est coûteux par rapport à envoyer un message de plus grande taille.*
Requête : *analyse syntaxique, sémantique, plan d'accès pour chaque nouvelle requête.*
Application : *conserver la connexion au SGBD, une requête retournant beaucoup de tuples et une boucle dans le programme est plus efficace que plusieurs petites requêtes.*

P4: Faire sur le serveur ce qui doit être fait sur le serveur

- Il faut déterminer la répartition des tâches entre client (programme d'application) et serveur (SGBD).
- Trois facteurs
  - Ressources respectives du client et du serveur.

Example 11 *Si le serveur est surchargé, on utilise si possible le client.*

P4: Faire sur le serveur ce qui doit être fait sur le serveur

- Il faut déterminer la répartition des tâches entre client (programme d'application) et serveur (SGBD).
- Trois facteurs
  - Ressources respectives du client et du serveur.

Example 11 *Si le serveur est surchargé, on utilise si possible le client.*

  - Localisation des données.

Example 12 *Pas clair Rafraîchissement d'un écran lors de mise à jour de la BD à l'aide de triggers (serveur) ou par des requêtes à intervalle régulier (client). Le choix des triggers est préférable.*

P4: Faire sur le serveur ce qui doit être fait sur le serveur

- Il faut déterminer la répartition des tâches entre client (programme d'application) et serveur (SGBD).
- Trois facteurs
  - Ressources respectives du client et du serveur.

Example 11 *Si le serveur est surchargé, on utilise si possible le client.*

  - Localisation des données.

Example 12 *Pas clair Rafraîchissement d'un écran lors de mise à jour de la BD à l'aide de triggers (serveur) ou par des requêtes à intervalle régulier (client). Le choix des triggers est préférable.*

  - Interaction avec l'utilisateur.

Example 13 *Éviter les interactions durant une transaction. Suppose lock*

P5: Etre prêt à faire des compromis

- Les différentes ressources sont étroitement liées donc une changement de l'une d'elles a des répercussions sur les autres
- Tout ce que l'on fait en tuning est une question de compromis
- *Vous voulez de meilleurs performances. Combien êtes vous prêts à payer pour cela ?*

Example 14

Plus de matériel $\Rightarrow$ *plus cher.*
Ajouter un index $\Rightarrow$ *plus de place occupée, plus de temps processeur utilisé pour les mises à jour.*