PRÉCONISATIONS POUR VOS APPLICATIONS 4D

Dernière mise à jour : 27 septembre 2019

Auteur : Cédric Gareau

SOMMAIRE

I- Introduction

Voici quelques préconisations et bonnes pratiques, pleines de bon sens, à suivre lorsque vous déployez des applications 4D, agrémentées d'informations techniques.

II- Applications 4D

Recommandé :

- Utiliser le mode Unicode
- Utiliser une version 64 bits
- Déployer une application 4D compilée ou exécutable en prenant soin de cocher au préalable, dans les propriétés de votre base de données, de :
    - cocher « Contrôle d'exécution »
    - cocher « Générer le fichier de symboles »
    - cocher « Générer le fichier d'erreur »
    - cocher « Compilation multi-cible (32 bits et 64 bits) »
    - sélectionner « Toutes les variables sont typées » pour le chemin de compilation
    - sélectionner « Entier long » pour les numériques
    - sélectionner « Entier long » pour les boutons
- Utiliser des champs UUID pour les clés primaires
- Dessiner les liens entre les tables
- Ne plus utiliser la commande « MODIFER SELECTION »
- Utiliser les objets
- Utiliser des listbox pour les listes
- Utiliser exclusivement des noms d'objets
- Dans l'onglet « Compatibilité » des propriétés de votre base de données convertie (issue d'une version précédente de 4D), cocher les options :
    - « Exécuter CHERCHER PAR FORMULE » sur le serveur et « Exécuter TRIER PAR FORMULE sur le serveur »
    - « CHERCHER PAR FORMULE utilise jointures SQL »

III- Numéros de port

Recommandé : s'assurer que les ports ci-dessous soient disponibles et dédiés à l'application 4D Server

- ports obligatoires :
    - port de publication client/serveur (modifiable dans les propriétés de la base, par défaut 19813)
    - port applicatif (non modifiable : port de publication +1, par défaut 19814)
    - port pour le serveur SQL (modifiable dans les propriétés de la base, par défaut 19812 ; même si vous n'utilisez pas le langage SQL, 4D vérifiera au démarrage si ce port est libre)
- ports facultatifs :
    - ports pour le serveur Web, utilisés pour les requêtes Web, SOAP ou REST (modifiables dans les propriétés de la base, par défaut 80 (HTTP) et 443 (HTTPS))

- port pour l'interpréteur PHP (modifiable dans les propriétés de la base, par défaut 8002 sur l'adresse 127.0.0.1)
- d'autres ports peuvent également être utilisés par 4D, notamment par le plugin 4D Internet Commands

Vous pouvez modifier les ports utilisés par 4D, par défaut. Cependant attention certains ports sont utilisés ou réservés pour d'autres applications :

- 0 à 1023 (Ports réservés) : Ces ports sont affectés par l'I.A.N.A. (Internet Assigned Numbers Authority) et sur la plupart des systèmes ne peuvent être utilisés que par des process système (ou racine) ou par des programmes exécutés par des utilisateurs disposant de privilèges d'accès avancés.
  - 20 et 21 FTP;
  - 23 TELNET;
  - 25 SMTP;
  - 37 NTP;
  - 80 et 8080 HTTP;
  - 443 HTTPS.

- 1024 à 49151 (Ports enregistrés) : Ces ports sont enregistrés par l'I.A.N.A. et peuvent être utilisés sur la plupart des systèmes par des process utilisateurs ou par des programmes exécutés par des utilisateurs sans privilèges particuliers (routeurs, applications spécifiques...)

- 49152 à 65535 (Ports dynamiques et/ou privés) : Ces ports sont d'utilisation libre.

Les personnes souhaitant utiliser les commandes TCP/IP pour synchroniser des bases de données doivent utiliser des numéros de port supérieurs à 49151.

Pour de plus amples informations, veuillez visiter le site Web de l'I.A.N.A. : http://www.iana.org

IV- Processeur

Recommandé :

- pour les postes de travail : 1 CPU avec 2 cœurs minimum

- pour les serveurs : 1 CPU avec 4 cœurs minimum (plutôt que 2 CPU avec 2 cœurs car la mémoire cache sera partagée entre tous les cœurs)

- utiliser le mode préemptif pour tirer intégralement parti des machines multi-cœurs (disponible à partir de la version 4D v15 R5 - exécuté en compilé 64 bits uniquement)

- déclarer explicitement toutes les méthodes que vous souhaitez démarrer en mode préemptif (cocher l'option dans les propriétés de vos méthodes et vérifier leur éligibilité grâce au compilateur)

Depuis la version 4D v11 SQL, 4D tire partie à travers le système des multiples cœurs (*). En effet, le système d'exploitation répartit le « temps processeur » (total du temps de tous les cœurs) entre chaque application et entre les threads de chaque application. C'est ensuite l'application qui définit les priorités de chacun de ses threads, chaque thread travaillant individuellement.

** : Les systèmes supportant le multi-threading permettent de simuler 2 cœurs logiques pour chaque cœur, multipliant ainsi par deux la capacité de traitement de 4D Server.*

Dans 4D, il existe des threads coopératifs et des threads préemptifs. 4D les utilise automatiquement (pas de logiciel ou de préférence spécifique) en fonction du code 4D exécuté.
Tous les threads coopératifs fonctionnent dans le thread principal à la différence des threads préemptifs qui sont dispatchés par le serveur 4D sur les autres threads. En effet, lorsqu'il est exécuté en mode

préemptif, un process est dédié à un CPU (processeur). La gestion du process est alors déléguée au système, qui peut allouer chaque CPU séparément sur une machine multi-cœurs.

Lorsqu'ils sont exécutés en mode coopératif (seul mode disponible dans 4D jusqu'à 4D v15 R5), tous les process sont gérés par le thread (process système) de l'application parente et partagent le même CPU, même sur une machine multi-cœurs.

Par conséquent, en mode préemptif, les performances globales de l'application sont améliorées, particulièrement avec des machines multi-cœurs, car de multiples threads peuvent véritablement être exécutés simultanément. Les gains effectifs dépendent cependant de la nature des opérations exécutées. Fondamentalement, le code destiné à être exécuté dans des threads préemptifs ne peut pas appeler d'éléments ayant des interactions extérieures telles que du code de plug-in ou des variables interprocess. L'accès aux données, cependant, est possible car le serveur de données de 4D prend en charge l'exécution en mode préemptif.

En contrepartie, puisqu'en mode préemptif chaque thread est indépendant des autres et non géré directement par l'application, des conditions spécifiques sont à respecter dans les méthodes qui doivent être exécutées en préemptif. De plus, le mode préemptif est disponible uniquement dans certains contextes.

Notes :

- *Un nouveau type de process, appelé process Worker, vous permet d'échanger des données entre n'importe quel process, y compris des process préemptifs.*

- *La nouvelle commande « APPELER FORMULAIRE » fournit une solution élégante permettant d'appeler des objets d'interface depuis un process préemptif.*

- *Bien qu'elles aient été conçues principalement pour les besoins liés à la communication interprocess dans le contexte des process préemptifs (accessibles en version 64 bits uniquement), les commandes « APPELER WORKER » et « APPELER FORMULAIRE » sont disponibles dans les versions 32 bits et peuvent être utilisées avec des process en mode coopératif.*

Il faut savoir aussi qu'un process non local de 4D Distant communique toujours avec deux threads jumeaux sur le serveur : un thread préemptif pour les requêtes DB4D (créer, stocker, charger, supprimer, trier, chercher, etc.) et un thread coopératif pour les requêtes applicatives (date du jour(*), lire variable process (-1;..), etc.). Un troisième thread préemptif peut aussi être créé si vous exécutez des commandes SQL (à l'appel à la commande « Debut SQL »).

En fonction de la commande exécutée, 4D utilise tel ou tel thread en établissant la communication sur le port correspondant :

- port de publication pour les requêtes DB4D (thread coopératif)
- port applicatif (port de publication +1) pour les requêtes applicatives (thread préemptif)
- port SQL pour les requêtes SQL (thread préemptif)

En conclusion, il est très fortement recommandé d'avoir un processeur multi-cœurs même si l'utilisation de tous les cœurs dépendra des commandes 4D utilisées dans votre application en sachant que plus vous êtes préemptif, plus vous serez rapide sur une machine multi-cœurs.

Important : si votre application 4D a été créée avec une très ancienne version de 4D, il est possible qu'une zone « Priorités CPU » soit visible dans l'onglet « Général » des propriétés de votre base de données. Ce paramétrage est désormais obsolète. Lorsque la zone est affichée, il est recommandé de cliquer sur le bouton « Réglages d'usine » afin de réinitialiser les paramètres et de les supprimer de la boîte de dialogue.

V- Système d'exploitation

Recommandé : OS 64 bits et certifié par 4D

Important : Déployer les applications 4D sur des systèmes d'exploitation certifiés par notre département Qualité. Les prérequis logiciel et système sont disponibles sur notre site web : https://fr.4d.com/resources/.

- **Sous Windows :**
  - Pour déployer l'application **4D** sur des postes de travail : **Windows 10**,
  - Pour déployer l'application **4D Server** sur un serveur ou l'application **4D** sur un serveur d'applications :
    - Windows **Server 2016**,
    - Windows **Server 2019** ;
  - Déployer l'application **4D Server** sur une machine dédiée (sous Windows Server, désactiver tous les rôles, notamment le rôle serveur de fichiers installé par défaut) ;
  - Déployer la gamme 64 bits :
    - **4D Server** est disponible en version 64 bits à partir de la version v12,
    - **4D** est disponible en version 64 bits à partir de la version v16 (en version Preview en 16.0 puis dans les versions 16 Rx) ;
  - Pour des raisons de performances, nous vous invitons également à désactiver la stratégie de sécurité locale "Client réseau Microsoft : communications signées numériques (lorsque le serveur l'accepte)" dans les options de sécurité, qui diminue de 15% les performances lorsqu'elle est active ;
  - Les paramètres par défaut des ordinateurs sous Windows et Windows Server sont optimisés pour économiser l'énergie. Il s'agit du meilleur réglage pour une utilisation bureau. Cependant un réglage sur « performances élevées » pour les serveurs permet d'obtenir des performances jusqu'à deux fois supérieures au mode « normal ».

- **Sous Mac :**
  - Pour déployer les applications **4D** ou **4D Server** :
    - macOS **High Sierra** (**10.13**),
    - macOS **Mojave** (**10.14**) ;
  - Déployer la gamme 64 bits :
    - **4D Server** est disponible en version 64 bits à partir de la version v15,
    - **4D** est disponible en version 64 bits à partir de la version v16.

- **Résolution d'écran :**
  - Les dialogues en 4D, tel que l'éditeur de recherches, nécessitent une résolution d'écran minimale de **1280 x 1024 pixels** ;
  - Selon le code de l'application 4D utilisé, les applications peuvent demander des résolutions plus petites (par exemple pour les appareils mobiles) ou plus grandes (par exemple, les grands écrans et les écrans à haute résolution).

Les versions 64 bits permettent aux applications 4D monopostes ainsi qu'aux applications 4D distantes de tirer pleinement parti des systèmes d'exploitation 64 bits. Le principal avantage de l'architecture 64 bits est qu'une mémoire de taille plus importante peut être adressée.

Bien que largement réécrites, les applications 4D 64 bits sont hautement compatibles avec les bases 4D courantes. Toutefois, étant donné qu'elles utilisent les technologies les plus récentes, nous avons dû mettre à jour quelques fonctions, et en arrêter d'autres.

D'un autre côté, l'implémentation de l'architecture 64 bits nous a donné l'opportunité de prendre en charge des fonctionnalités puissantes comme les process 4D préemptifs (multithread), de moderniser les impressions ainsi que les d'éditeurs d'états rapides et d'étiquettes, ou encore de faire bénéficier vos applications des animations d'objets natives (4D 64 bits sous OS X).

VI- Mémoire

Recommandé :

- pour les postes de travail : 8 Go minimum
- pour les serveurs : 16 Go minimum
- ECC (technologie de correction d'erreurs)

La mémoire allouée à l'application 4D Server dépend de 4 facteurs :

- la quantité de mémoire totale sur votre machine,
- la quantité de mémoire disponible,
- du système d'exploitation (32 ou 64 bits),
- de la version de 4D Server utilisée (32 ou 64 bits).

Pour information, il n'est pas possible d'avoir un cache inférieur ou égal à 100 Mo. De toute façon le cache est très important, il faut donc réserver une quantité de mémoire suffisante (après calcul de la mémoire nécessaire pour les process, etc.) au cache de 4D.

- L'espace mémoire réservé au cache est séparé de l'espace mémoire utilisé par le serveur pour les process. Effectivement le fait d'augmenter la taille du cache diminue l'espace réservé aux process.
- La taille maximale varie en fonction du nombre d'utilisateurs connectés et du nombre de process par utilisateur. Cependant 1 Go environ est par défaut utilisé par 4D (en dehors des process), notamment pour charger toutes les librairies systèmes.
- La gestion du cache a été améliorée. En particulier, un nouveau mécanisme effectue les opérations les plus consommatrices dans la mémoire temporaire, ce qui permet d'alléger le cache principal. La mémoire temporaire a pour avantage de n'être utilisée qu'en cas de besoin et ne mobilise pas les ressources de la machine. Nous vous invitons à calculer la mémoire nécessaire au bon fonctionnement de 4D Server, et ensuite de la déduire de la mémoire totale allouée par le système à 4D. Vous pourrez ainsi en déduire la taille de cache maximale que vous pouvez allouer. Une taille de cache excessive risque d'ailleurs de diminuer les performances générales du système, voire de le rendre instable.
- Pour connaître le taux de réussite il faut observer les variations de la mémoire cache observée. Lorsque vous observez une diminution du cache utilisé, cela signifie que 4D a eu besoin de supprimer des objets afin de libérer de la mémoire pour certains objets du moteur de données. Si le cache est purgé encore et encore, c'est une bonne analyse qui indique que le cache est trop petit. Dans ce cas, 4D a besoin de purger de manière répétée un quart de son cache dans le but de libérer assez de place pour les objets du moteur de données.

- La gestion du cache doit être laissé aux soins de 4D, seules la taille du cache et la fréquence d'écriture du cache sont importantes désormais. Nous vous conseillons de ne pas utiliser la commande « ECRIRE CACHE » par exemple, il est préférable d'utiliser l'option « Ecriture cache toutes les 20 secondes », qui spécifie les intervalles de sauvegarde des données, afin de contrôler l'écriture du cache de données sur le disque. 4D utilise en interne un système intégré de cache de données permettant d'accélérer les opérations d'Entrée/Sortie. Le fait que des modifications de données soient, par moment, présentes dans le cache de données et pas sur le disque, est entièrement transparent pour votre code. Par exemple, si vous appelez la commande CHERCHER, le moteur de 4D va intégrer les données présentes dans le cache pour effectuer l'opération.

Nous vous conseillons :

- de ne pas cocher l'option « Calcul du cache adaptatif » afin de fixer une taille de cache fixe ;
- de décocher, sous Mac, l'option « Maintenir le cache en mémoire physique » ;
- prévoir suffisamment de mémoire dans la machine pour le cache, la mémoire moteur de 4D Server et le système d'exploitation lui-même ;
- déployer la version 64 bits de 4D Server (une application 32 bits ne peut pas utiliser plus de 4 Go de mémoire, une application 64 bits dispose quant à elle de 8 To d'espace adressable théorique !) ;
- déterminer la valeur idéale du cache pour votre base en production en utilisant le composant « 4D_Info_Report » (http://taow.4d.com/Outil-4D-Info-Report/PS.1938271.fr.html).

(Prévoir des slots de libre si le fichier de données est amené à grossir rapidement pour rajouter de la mémoire par la suite)

Le gestionnaire du cache de la base de données a été entièrement réécrit en 4D v16 et améliore ainsi l'utilisation d'un cache très important pour les ordinateurs modernes (avec 64 ou même 128 Go de cache) permettant de profiter des faibles prix des barrettes mémoire et permettant ainsi de stocker une base de données de grande taille entièrement en mémoire. Il améliore également les situations où le cache est de petite taille alors que le fichier de données est très volumineux grâce à une meilleure gestion des priorités pour les objets de données à contenir ou à libérer du cache.

En conséquence, la base de données sera plus rapide, permettant de gérer plus de données et plus d'accès utilisateurs simultanés.

Enfin, depuis la v16, le cache peut être configuré ou analysé dynamiquement avec les commandes suivantes :

- La commande « ECRIRE CACHE » accepte désormais un paramètre * pour vider le cache ou un nombre d'octets minimum de libération du cache (uniquement pour effectuer des tests)
- La commande « FIXER TAILLE CACHE » fixe dynamiquement la taille du cache de la base de données dans les versions 64 bits de 4D
- La commande « Lire informations cache » récupère des informations relatives à l'utilisation du cache en 64 bits
- La commande « Lire taille cache » retourne la taille courante du cache
- Le sélecteur « Périodicité écriture cache » de la commande « FIXER PARAMETRE BASE » permet de lire ou de fixer la périodicité de l'écriture du cache sur le disque

Dans les versions 64 bits de 4D, le cache des données de la base inclut un mécanisme de gestion automatique des priorités offrant un haut niveau d'efficacité et de performance. Ce mécanisme permet d'optimiser la rotation des données dans le cache lorsque le programme a besoin de place : les données de plus faible priorité sont déchargées en premier, tandis que les données de priorité plus haute restent chargées.

Ce mécanisme est entièrement automatique et la plupart du temps, vous n'aurez pas besoin de vous en préoccuper. Cependant, pour des cas particuliers, il peut être personnalisé à l'aide d'un ensemble de commandes dédiées, vous permettant de changer la priorité des objets pour toute la session ou uniquement le process courant. A noter que ces commandes doivent être utilisées avec précaution car elles peuvent affecter les performances de la base.

Le gestionnaire du cache sélectionne les données à retirer du cache en cas de besoin à l'aide d'un système de priorité. Les trois types d'objets qui peuvent être chargés dans le cache ont une priorité différente :

- tables : toutes les données standard des champs (numériques, dates...), à l'exclusion des blobs (voir ci-dessous). Priorité par défaut : moyenne

- blobs : toutes les données binaires des champs (textes, images, objets et blob) stockées dans le fichier de données. Priorité par défaut : faible

- index : tous les index de champs simples, y compris les index de mots-clés et les index composites. Comme les index sont utilisés très fréquemment, ils ont un statut spécial dans le cache. Priorité par défaut : élevée

Les priorités par défaut assurent généralement des performances optimales. Cependant, dans certains cas spécifiques, vous pouvez avoir besoin de personnaliser ces priorités. Pour cela, vous disposez de deux ensembles de commandes 4D :

- Les commandes qui modifient les priorités du cache pour l'ensemble de la session et tous les process : « FIXER PRIORITE CACHE TABLE », « FIXER PRIORITE CACHE INDEX » et « FIXER PRIORITE CACHE BLOBS ». Ces commandes doivent appelées au démarrage de la base.

- Les commandes qui modifient les priorités du cache pour le process courant uniquement : « AJUSTER PRIORITE CACHE TABLE », « AJUSTER PRIORITE CACHE INDEX » et « AJUSTER PRIORITE CACHE BLOBS ». Utilisez ces commandes si vous souhaitez changer temporairement la priorité des objets dans le cache afin d'améliorer les performances lors d'une opération temporaire, puis revenir aux priorités initiales.

Ces commandes sont disponibles uniquement dans les contextes suivants :

- 4D version 64 bits

- 4D Server ou 4D en mode local

VII- Disque dur

Recommandé :

- Un système de disques SSD ultra performant avec RAID 10 matériel (avec une carte contrôleur RAID, de type PERC par exemple) pour stocker la base de données 4D

- Un disque SSD de secours (si un des disques du système RAID tombe en panne)

- Un disque de capacité suffisante pour le fichier d'historique courant et les sauvegardes (4BK et 4BL)

- Restituer de temps en temps la dernière sauvegarde (4BK) sur une autre machine et effectuer une vérification des données avec l'outil intégré (CSM) dans 4D. Cette action peut être automatisée (nécessite un peu de développement)

Attention : il est préférable de s'intéresser aux vitesses d'écriture et de lecture avant d'acheter un disque SSD.

L'idéal serait de pouvoir y stocker le système, 4D Server et l'intégralité de la base de données.

Ne négligez pas l'achat d'un disque de secours (pour garantir la protection du système RAID) :

- Lorsqu'un disque est défectueux : il n'y a plus aucune protection tant que ce disque n'est pas réparé.

- Lorsque deux disques sont défectueux : le système s'arrête.

Cela signifie :

- qu'il faut surveiller le disque, et ne pas faire une confiance aveugle au RAID,

- que ce n'est pas le jour où vous êtes confronté à un souci qu'il faut commander le disque de remplacement. En effet, le chef capable d'acheter sera alors indisponible, voir le produit sera en rupture de stock chez le fournisseur… Bref, vous vous retrouverez alors dans une situation où un second disque de la même marque, de la même série et probablement du même lot n'aura qu'une envie : se mettre au repos comme son petit frère !

Pour améliorer la tolérance aux pannes, la sécurité et/ou les performances de l'ensemble, la mise en place d'un système RAID est un très bon choix :

- pour la base de données : le meilleur choix est le RAID 10 (sécurité et performances),

- pour les sauvegardes : le meilleur choix est le RAID 5 (prix et sécurité).

Vous trouverez ci-dessous un comparatif des différents niveaux de RAID suivi des types de système RAID. Ce tableau n'a pas été réalisé par 4D mais je trouve qu'il résume bien ce qu'il faut savoir sur le RAID.

Fonctionnalités	RAID 0	RAID 1	RAID 1E	RAID 5	RAID 5EE	RAID 6	RAID 10	RAID 50	RAID 60
Nbre minimum de disques	2	2	3	3	4	4	4	6	8
Protection des données	Pas de protection	Panne d'un seul disque	Panne d'un seul disque	Panne d'un seul disque	Panne d'un seul disque	Panne de deux disques	Une panne de disque maxi dans chaque sous-pile	Une panne de disque maxi dans chaque sous-pile	Deux pannes de disque maxi dans chaque sous-pile
Performances en lecture	Elevées	Elevées	Elevées	Elevées	Elevées	Elevées	Elevées	Elevées	Elevées
Performances en écriture	Elevées	Moyennes	Moyennes	Faibles	Faibles	Faibles	Moyennes	Moyennes	Moyennes
Performances en lecture (mode dégradé)	S/O	Moyennes	Elevées	Faibles	Faibles	Faibles	Elevées	Moyennes	Moyennes
Performances en écriture (mode dégradé)	S/O	Elevées	Elevées	Faibles	Faibles	Faibles	Elevées	Moyennes	Faibles
Utilisation de la capacité	100 %	50 %	50 %	67 % - 94 %	50 % - 88 %	50 % - 88 %	50 %	67 % - 94 %	50 % - 88 %
Applications types	Stations de travail de haut de gamme, journalisation de données, rendu en temps réel, données très transitoires	Système d'exploitation, bases de données transactionnelles	Système d'exploitation, bases de données transactionnelles	Entreposage de données, mise en oeuvre de serveurs Web, archivage	Entreposage de données, mise en oeuvre de serveurs Web, archivage	Archivage de données, sauvegarde sur disque, solutions à haute disponibilité, serveurs gourmands en capacité	Base de données à accès rapide, serveurs d'applications	Bases de données volumineuses, serveurs de fichiers, serveurs d'applications	Archivage de données, sauvegarde sur disque, solutions à haute disponibilité, serveurs gourmands en capacité

	Logiciels	Matériels	Matériels externes
Description	Idéal pour les applications basées sur des blocs de grande taille, comme l'entreposage de données ou la vidéotransmission en continu. Convient aussi dans les cas où les serveurs disposent des cycles UC qui leur permettent de gérer les opérations gourmandes en E/S requises par certains niveaux RAID. Inclus dans le système d'exploitation, comme Windows®, Netware et Linux. Toutes les fonctions RAID sont gérées par l'unité centrale de l'hôte, ce qui peut gravement amputer sa capacité à effectuer d'autres calculs.	Idéal pour les applications basées sur des blocs de petite taille, comme les bases de données transactionnelles et les serveurs Web. Les opérations RAID gourmandes en temps processeur sont déchargées de l'unité centrale de l'hôte pour optimiser les performances. L'utilisation d'un cache à écriture différée avec batterie de secours peut améliorer les performances de manière considérable sans risque de perte de données.	La connexion au serveur est établie via un contrôleur standard. Les fonctions RAID sont exécutées sur un microprocesseur situé sur le contrôleur RAID externe indépendant de l'hôte.
Avantages	Économique Nécessite seulement un contrôleur standard	Avantages de RAID en matière de protection des données et de performances Fonctionnalités de tolérance aux pannes plus robustes et performances optimisées par rapport au système RAID logiciel	Indépendant du système d'exploitation Permet de développer des systèmes de sauvegarde de grande capacité pour les serveurs de haut de gamme

VIII-Sauvegarde / Opérations de maintenance

<u>Recommandé :</u>

- Dans les propriétés de la base, onglet « Périodicité » :
  - Activer la sauvegarde automatique de 4D (une fois par jour)

- Dans les propriétés de la base, onglet « Configuration » :
  - Cocher « Fichier de données »
  - Cocher « Fichier de structure »
  - Cliquer sur le bouton « Ajouter fichier » et sélectionner :
    - le fichier d'index des données « .4DIndx »
    - les 2 fichiers « .4DSyncData » et « .4DSyncHeader » (si vous utilisez le mécanisme de réplication de 4D)
  - Renseigner l'emplacement des fichiers de sauvegarde et vérifier que l'espace libre est suffisant pour stocker tous les fichiers de sauvegarde
  - Cocher l'option « Utiliser le fichier d'historique »

- Dans les propriétés de la base, onglet « Sauvegarde & restitution » :
  - Cocher l'option « Conserver les X derniers fichiers de sauvegarde »
  - Choisir l'option « Effacer la sauvegarde la plus ancienne après sauvegarde »
  - Cocher « Réessayer dans 60 secondes »
  - Décocher l'option « Annuler l'opération au bout de X tentatives »
  - Cocher l'option « Restituer la dernière sauvegarde si la base est endommagée »
  - Cocher l'option « Intégrer le dernier historique si la base est incomplète »

- Vérifier et compacter le fichier de données régulièrement (depuis la v13, il est possible de détecter la fragmentation d'une table 4D et donc d'agir en conséquence grâce à la commande « Lire fragmentation table »)

- Fermer la fenêtre d'administration après chaque utilisation

Depuis la version 4D v15 R4, nous avons optimisé de façon importante l'algorithme de réindexation globale de la base de données. Tout le processus a été revu, et l'opération peut s'effectuer désormais jusqu'à deux fois plus rapidement.

Note : Une réindexation globale est nécessaire, par exemple, après une réparation de la base de données ou lorsque le fichier .4dindx a été supprimé.

Comme chaque enregistrement de chaque table indexée doit être chargé en mémoire durant l'indexation, l'optimisation a visé à minimiser les échanges entre le cache et le disque (swaps). L'opération est

désormais effectuée séquentiellement sur chaque table, ce qui réduit les besoins en chargement et en déchargement d'enregistrements.
Idéalement, si le cache était assez grand pour contenir la totalité du fichier de données et des index, le nouvel algorithme de réindexation n'apporterait aucune amélioration. Cependant, la mémoire disponible sur le serveur n'est généralement pas aussi grande. Si le cache est assez grand pour contenir au moins les données et les index de la table la plus volumineuse, alors le nouvel algorithme sera jusqu'à deux fois plus rapide que le précédent.

Réaliser des sauvegardes régulières des données est important mais ne permet pas, en cas d'incident, de récupérer les données saisies depuis la dernière sauvegarde. Pour répondre à ce besoin, 4D dispose d'un outil particulier : le fichier d'historique. Ce fichier permet d'assurer la sécurité permanente des données de la base.
En outre, 4D travaille en permanence avec un cache de données situé en mémoire. Toute modification effectuée sur les données de la base est stockée provisoirement dans le cache avant d'être écrite sur le disque dur. Ce principe permet d'accélérer le fonctionnement des applications ; en effet, les accès mémoire sont bien plus rapides que les accès disque. Si un incident survient sur la base avant que les données stockées dans le cache aient pu être écrites sur le disque, vous devrez intégrer le fichier d'historique courant afin de récupérer entièrement la base.

Si vous travaillez en environnement virtuel, il est recommandé d'arrêter la base avant d'effectuer un snapshot, pour être certain que toutes les données stockées en mémoire soient écrites dans le fichier de données.

En plus de la sauvegarde et du fichier d'historique de 4D, nous vous invitons à planifier régulièrement des opérations de maintenance en vérifiant et en compactant le fichier de données et d'index.

Une fois votre stratégie de sauvegarde mise en place, nous vous invitons à envisager le pire (incendie, vol) et donc d'effectuer une copie hebdomadaire de la base sur un support inerte dans un autre endroit sécurisé.

Dans le cadre d'applications critiques, il est également possible de mettre en place un système de sauvegarde par miroir logique, permettant un redémarrage instantané en cas d'incident sur la base en exploitation. Les deux machines communiquent par le réseau, la machine en exploitation transmettant régulièrement à la machine miroir les évolutions de la base par l'intermédiaire du fichier d'historique. De cette façon, en cas d'incident sur la base en exploitation, vous pouvez repartir de la base miroir pour reprendre très rapidement l'exploitation sans aucune perte de données.

Les principes mis en œuvre sont les suivants :

- La base de données est installée sur le poste 4D Server principal (poste en exploitation) et une copie identique de la base est installée sur le poste 4D Server miroir.

- Un test au démarrage de l'application (par exemple la présence d'un fichier spécifique dans un sous-dossier de l'application 4D Server) permet de distinguer chaque version (en exploitation et en miroir) et donc d'exécuter les opérations appropriées.

- Sur le poste 4D Server en exploitation, le fichier d'historique est segmenté à intervalle régulier à l'aide de la commande « Nouveau fichier historique ». Aucune sauvegarde n'étant effectuée sur le serveur principal, la base de données est en permanence disponible en lecture/écriture.

- Chaque segment de fichier d'historique est envoyé sur le poste miroir, où il est intégré à la base miroir à l'aide de la commande « INTEGRER FICHIER HISTORIQUE ».

La mise en place de ce système nécessite la programmation de code spécifique, notamment :

- un minuteur sur le serveur principal pour la gestion des cycles d'exécution de la commande « Nouveau fichier historique »,

- un système de transfert des segments de fichier d'historique entre le poste en exploitation et le poste miroir (utilisation de 4D Internet Commands pour un transfert via ftp ou messagerie, Web Services, etc.),

- un process sur le poste miroir destiné à superviser l'arrivée de nouveaux « segments » de fichier d'historique et à les intégrer via la commande « INTEGRER FICHIER HISTORIQUE »,

- un système de communication et de gestion d'erreurs entre le serveur principal et le serveur miroir.

Attention : La sauvegarde par miroir logique est incompatible avec les sauvegardes « standard » sur la base en exploitation car l'emploi simultané de ces deux modes de sauvegarde entraîne la désynchronisation de la base en exploitation et de la base miroir. Par conséquent, vous devez veiller à ce qu'aucune sauvegarde, automatique ou manuelle, ne soit effectuée sur la base en exploitation. En revanche, il est possible de sauvegarder la base miroir.

Depuis la version v14, il est possible d'activer le fichier d'historique courant sur le poste miroir. Vous pouvez ainsi mettre en place un « miroir de miroir », ou des serveurs miroirs en série. Cette possibilité s'appuie sur la commande « INTEGRER FICHIER HISTORIQUE MIROIR ».

IX- Réseau

Recommandé :

- Utiliser l'ancienne couche réseau jusqu'à la version v15 R5

- Utiliser la nouvelle couche réseau à partir de la version 16.2 publique

Attention : l'ancienne couche réseau n'est pas disponible dans les versions 64 bits de 4D Developer (Windows et Mac) et dans la version 64 bits de 4D Server (Mac uniquement).

Il est très important pour 4D Server d'avoir en permanence suffisamment de bande passante pour communiquer avec ses postes distants. Si vous mutualisez la bande passante, il faudra en réserver une partie pour 4D Server.
En effet, lorsque la bande passante vient à manquer, certains paquets sont perdus et cela provoque inévitablement des erreurs côté Serveur. Si trop d'erreurs réseau surviennent au même moment ou de façon trop fréquence, vous déstabilisez 4D Server.

Sachez également que, si vous utilisez le serveur Web de 4D et que vous désirez séparer le trafic pour des raisons de sécurité et/ou de performances, il est possible de dédier une carte réseau aux utilisateurs de 4D Distant et une autre aux requêtes Web, SOAP ou REST.

X- Web

Recommandé :

- utiliser une version 64 bits de 4D
- utiliser 4D Server ou 4D en mode local (le mode préemptif n'est pas pris en charge par 4D en mode distant)
- déployer une application 4D compilée ou exécutable
- avoir le maximum de méthodes bases et méthodes projets relatives au Web confirmées thread-safe par 4D Compiler
- dans les propriétés de votre base :
    - cocher l'option « Utiliser des process préemptifs » pour activer le mode préemptif,
    - cocher l'option « utiliser le cache Web de 4D » et fixer une taille de cache de 524 288 Ko,
    - fixer à 8 heures le délai de conservation des process inactifs et cocher la case « gestion automatique des sessions » pour que les utilisateurs Web puissent réutiliser le même contexte durant la journée (si vous ne les gérez pas par programmation),
    - cocher l'option « utiliser les connexions persistantes ».

Depuis la version v16, le serveur Web intégré de 4D (en 64 bits uniquement) pour Windows et pour Mac OS X permet de tirer pleinement parti du multi-cœurs en utilisant des process Web préemptifs dans les applications compilées. La plupart des commandes de 4D liées au Web, les méthodes et les URL de la base de données sont thread-safe et peuvent être utilisées en mode préemptif. Vous pouvez configurer votre code lié au Web, y compris les balises HTML 4D et les méthodes base Web, afin qu'il s'exécute simultanément sur le plus grand nombre de cœurs possibles.

XI- Machine physique ou virtuelle

Recommandé : machine physique

Même si 4D fonctionne en environnement virtuel et est donc éligible en terme d'exploitabilité, côté performances notre expérience nous montre qu'une machine physique offre de meilleures performances dans le temps à ressources équivalentes.

S'il ne vous est pas imposé de virtualiser le serveur 4D en production, nous vous conseillons dans un premier temps de le déployer sur une machine physique. Puis dans un second temps, de programmer des tests poussés en environnement virtuel. Vous aurez ainsi l'avantage de pouvoir comparer les 2 solutions.

Toutefois, si l'on peut trouver un avantage à la virtualisation c'est la souplesse d'allocation des ressources (CPU, mémoire notamment). Il est possible d'allouer des ressources supplémentaires par la suite, voir même d'allouer des ressources en temps réel, en fonction de l'activité de la machine. Nous avons d'ailleurs un certain nombre de clients qui travaillent avec des environnements virtualisés, de plus en plus d'ailleurs. Nous avons même des clients qui déploient des serveurs TSE virtualisés sur des serveurs lames.

Nous n'avons cependant pas de documents officiels certifiant le fonctionnement de 4D en environnement virtuel ou privilégiant une solution plutôt qu'une autre.

Nous préconisons uniquement de s'assurer que les performances de la machine soient suffisantes en termes de ressources (mémoire, processeur, etc.) ou que le système d'exploitation virtualisé soit certifié avec la version de 4D installée.
De manière générale les ressources CPU et RAM doivent être un peu plus importantes en environnement virtualisé par rapport à une solution non virtualisée.

De plus les performances observées dépendent grandement du paramétrage de la machine virtuelle (CPU, mémoire, etc. mais aussi du disque dur et de la carte réseau (caractéristiques, est-elle partagée, dédiée, correctement configurée, etc.)), de la solution utilisée, de la version de la solution et du système sur lequel est installé la solution. Pour cette partie je vous invite à consulter les sites et forums des éditeurs de solutions virtualisées ainsi que les études comparatives.

Remarque : Si vous travaillez en environnement virtuel, il est recommandé d'arrêter la base avant d'effectuer un snapshot, pour être certain que toutes les données stockées en mémoire soient écrites dans le fichier de données.

XII- Marque / Modèle de machine

Nous n'avons pas de préconisations particulières, il faut cependant regarder en détails le matériel qui compose la machine avant de l'acheter afin de s'assurer que les composants soient compatibles entre eux et que ses caractéristiques correspondent à vos attentes et aux préconisations ci-dessus.

XIII-Tests / Recette / Déploiement

Ne déployez pas votre base de données sans l'avoir testé au préalable dans son futur environnement ou dans un environnement similaire (matériel, version de 4D identique, etc.) et surtout dans ses futures conditions d'utilisation (avec le même nombre d'utilisateurs simultanés en utilisant des scénarios de tests Utilisateur).
Stresser sa base de données pour en connaître les limites, non détectables par l'équipe de développement, vous épargnera bien des soucis en Production et permettra d'optimiser les fonctionnalités les plus utilisées.

Les tests et la recette sont les clés d'un déploiement réussi !

XIV-Sécurité

Source : https://blog.4d.com/security-data-protection/

La sécurité est un sujet important et fondamental pour une base de données ou un système de solution entreprise. Ce chapitre propose un aperçu de la façon dont 4D protège vos données. En fait, la sécurité est la protection des données. Et la protection des données est une vaste zone. Les données doivent être protégées pour les accès indésirables, mais aussi pour la perte. Ceci est un fait important, car la plupart des utilisateurs ne pensent qu'à la protection des utilisateurs non autorisés, pas de protection pour des événements tels que panne de courant, disque dur endommagé, les modifications accidentelles de données et ainsi de suite.

Sécurité et protection des données est une zone très large : cela commence par l'authentification de l'utilisateur, passe par l'accès externe (comme le Web ou le SQL), l'exécution de code indésirable (injection SQL, attaques d'inspection du script), puis les mises à jour de sécurité, de sauvegarde et plus encore.

4D Server

4D Server est un système intégré de développement client / serveur, optimisé pour créer des applications d'entreprise robustes avec un système de base de données intégrée. Bien que 4D peut envoyer des données (avec des normes telles que HTTP, SOAP, ODBC ou OCI) ou peut être accessible à partir de l'extérieur (avec HTTP, SOAP, ODBC / SQL), l'utilisation principale est basée sur le langage de développement interne « 4D », en utilisant un protocole réseau interne et propriétaire pour communiquer entre le client et le serveur.

La communication réseau prend en charge le cryptage TLS 1.2, soit à l'aide d'une clé prédéfinie (pas de certificat SSL requis), soit avec un fichier contenant une clé.

La liaison étroite entre le langage de développement et la communication réseau permet une construction de haut niveau dans le concept de protection, en évitant les scénarios d'attaque typiques tels que l'injection SQL ou des attaques « buffer overflow ».

Le langage 4D est un langage puissant et mature, parfaitement conçu pour construire des systèmes d'applications d'entreprise. Il propose plus de 1 500 commandes, couvrant les opérations de base de données (tris, requêtes, créations, transactions et ainsi de suite), l'impression, la communication avec d'autres appareils ou ordinateurs, la gestion des documents, une fenêtre ou une interface utilisateur, et bien plus encore. Jeter un coup d'œil au manuel du langage 4D pour plus de détails.

Le langage lui-même est segmenté, même en mode interprété (développement ou prototypage), il n'est jamais exécuté comme une évaluation de texte. En mode production, le langage est compilé et intègre un contrôle automatique de plage de version contre les attaques de type « buffer overflow ».

Serveur Web

4D dispose de son propre serveur HTTP, un puissant serveur multithread pour les contenus statiques et dynamiques. L'intégration étroite a un impact considérable sur la sécurité accrue.

Sans compter une meilleure sécurité du code (voir ci-dessous), ce concept supprime le problème de mise à jour typique oublié. Comme tout est intégré, il n'y a qu'un seul logiciel à mettre à jour. Les solutions habituelles nécessitent une énorme quantité de logiciels à maintenir à jour : PHP, OpenSSL, Apache, NodeJS, etc. Tous ont besoin de mises à jour régulières et il est commun que certaines parties restent non patchées pendant longtemps, en particulier si elles sont utilisées comme solution de service, sans une équipe informatique spécialisée.

Les requêtes web déclenchent du code 4D, qui répondent à la demande au niveau applicatif, pas seulement au niveau de la base de données. L'intégration étroite permet de contrôler toutes les requêtes, utilisant la construction d'autorisations et d'implémentations sur mesure, bien sûr crypté TLS.

Le serveur intégré HTTP permet également des justifications de contrôle fin, par exemple pour un serveur REST.

La version 4D v16 R6 prend désormais en charge Perfect Forward Secrecy (PFS). Cela vous donne le niveau de sécurité le plus élevé pour vos communications, par défaut ! Au delà de la protection qu'il offre, le soutien de PFS augmente également les résultats des tests de vérification SSL, ce qui est excellent pour nos clients. En particulier, ceux qui travaillent avec des informations sensibles.

Le niveau de sécurité par défaut du serveur Web de 4D a été augmenté pour être conforme à certaines fonctions de sécurité réseau (Sécurité App Transport (ATS) sur iOS, par exemple), et permet d'obtenir de meilleurs résultats lors des tests de vérification de sécurité Web (par exemple: SSL Labs).

Pour ce faire, nous avons :

- activé « Perfect Forward Secrecy », et
- désactivé l'algorithme RC4 de la liste de chiffrement.

Par conséquent, le serveur Web 4D obtient un « A » dans le classement de SSL Labs, sans qu'aucune action ne soit nécessaire !

Perfect Forward Secrecy (PFS) est un algorithme d'échange de clés. Il utilise les algorithmes Diffie-Hellman (DH) pour générer des clés de session de telle sorte que seuls les deux parties impliquées dans la communication peuvent les obtenir.

4D active automatiquement PFS lorsque TLS est activé sur le serveur. Pour cela, 4D génère un fichier « dhparams.pem » - si il n'existe pas déjà - qui contient la clé privée de votre serveur DH. Si vous utilisez la liste de chiffrement standard de 4D, PFS est prêt à l'emploi. Si vous préférez utiliser une liste de chiffrement personnalisé, vérifiez qu'il contient des entrées avec des algorithmes de ECDH ou DH.

Pour savoir si PFS est activé sur votre serveur Web, exécutez la commande « WEB Get server info » avec le nouvel attribut « perfectForwardSecrecy ». Cela permet de vérifier si toutes les conditions nécessaires pour utiliser PFS sont remplies :

- TLS est activé
- La liste de chiffrement contient au moins un algorithme ECDH ou DH
- Le fichier « Dhparams.pem » est présent et valide
- Tous les certificats SSL / TLS sont présents

L'algorithme RC4 a connu des problèmes de sécurité et est maintenant déprécié dans le serveur Web de 4D. Tous les chiffrements RC4 ont été retirés de la liste de chiffrement par défaut et le modèle « !RC4 » a été ajouté à la liste de chiffrement pour l'interdire explicitement.

Serveur SOAP

Similaire au serveur HTTP, un serveur SOAP est intégré, ce qui permet un contrôle détaillé d'accès, basé sur des objets métiers (pas seulement au niveau de la base de données).

Serveur SQL

Alors que l'accès aux données pour un client 4D utilise par défaut un protocole propriétaire, l'accès SQL (natif ou via ODBC) est aussi bien pris en charge. En outre, il existe des drivers PDO Open Source disponibles (PHP Data Objects). L'accès SQL au niveau base de données peut être contrôlé par un système de mot de passe, des schémas SQL ou à l'aide de vues SQL.

Système de mot de passe 4D

Le système propriétaire d'autorisation d'accès utilisateur de 4D peut être remplacé par des systèmes tiers. 4D prend en charge l'utilisation directe de Microsoft Active Directory et LDAP, ainsi que des systèmes entièrement personnalisés.

Mécanisme de mise à jour logicielle

Les logiciels modernes sont une combinaison complexe de logiciels, serveur de base de données, middleware, serveur d'applications, serveur web et bien plus encore. Il est facile d'oublier de garder toutes les pièces à jour, comme une DLL d'OpenSSL par exemple. 4D réduit ce problème à bien des égards, non seulement en aidant l'administrateur dans sa vie quotidienne, mais aussi en réduisant le risque par la conception.

Tout étant intégré en une seule solution, il n'y a qu'un seul dossier à remplacer. Tout est installé dans un seul dossier, il pourrait même être remplacé par un processus de glisser-déposer. Faire simple évite le syndrome « je le ferai plus tard ». Avec un seul remplacer, toutes les parties de l'application métier sont mises à jour en une seule étape, rien ne peut manquer.

Le serveur peut être mis à jour automatiquement. Le processus de mise à jour est contrôlé et forcé par 4D lui-même, il peut donc être piloté par le développeur de la solution.

Système de sauvegarde et de journalisation

4D fournit par défaut un système de journalisation transactionnelle. Chaque opération de modification de données est enregistrée et peut être annulée. En cas d'urgence, le travail de la journée peut être restaurée - rien n'est perdu. Dans le cas d'une interruption, la base de données est automatiquement vérifiée lors du redémarrage et les opérations manquantes (conservées en mémoire mais non stockées sur le disque encore) sont restaurées, afin d'avoir une base de données contenant toutes les informations. Même dans le cas d'une corruption totale des données (endommagement du disque, etc.), le fichier de données est automatiquement restauré à partir de la dernière sauvegarde complète et le fichier d'historique (contenant le travail quotidien) est intégré.

Le journal des transactions peut également être utile en cas de suppression accidentelle (ou volontaire d'enregistrements) et, à la fois pour la récupération légale des données.

La sauvegarde standard fait partie du produit 4D, aucune licence supplémentaire n'est nécessaire, seul un disque dur additionnel est vivement conseillé (pour se protéger des pannes de disque).

Dans les environnements 24/7, 4D prend en charge l'utilisation de systèmes miroirs montés en cascade et/ou en étoile. Une production, un miroir et un miroir secondaire permettent d'assurer un service 24 heures sur 24. Un système de miroir supplémentaire pourrait être exécuté dans une autre ville ou le Cloud pour protéger les données, même en cas de catastrophes extrêmes.

Parallèlement à cela, le système de journalisation transactionnelle de 4D prend en charge les snapshots des machines virtuelles (comme Volume Shadow Copy Service de VMWare vSphere (Hyperviseur ESXi, pris en charge à partir de la version 16 R2).

Protection additionnelle

Tous les concepts de protection standard, tels que la protection de la salle Serveur ou l'utilisation de disques durs cryptés (solutions matérielles avec SSD chiffré ou utilisation de logiciel crypté comme Bitlocker) sont bien sûr recommandés.

Voici également un lien vers notre guide sur la sécurité : https://blog.4d.com/4d-security-guide/

XV- Surveillance du serveur

Vous pouvez utiliser le composant « 4D_Info_Report » pour collecter un maximum d'informations :

- sur l'environnement système, matériel, 4D
- sur la base : structure, données, triggers, index, réglages personnalisés utilisés, etc.
- en temps réel : mémoire, cache, utilisateurs connectés, process, etc.

Ce composant peut être utilisé en Production sans problème : l'impact sur les performances de l'application monitorée est négligeable.

Vous pouvez également utiliser l'onglet « Moniteur » de la fenêtre d'administration de 4D Server pour afficher des informations dynamiques relatives à l'exploitation de la base de données ainsi que des informations sur le système et l'application 4D Server.

La zone graphique permet de visualiser l'évolution en temps réel de plusieurs paramètres :

- le taux d'utilisation des processeurs,

- le trafic réseau,
- l'état de la mémoire.

Ces informations peuvent être obtenues par programmation grâce à la commande « LIRE APERCU ACTIVITE ». Cette commande permet d'obtenir un instantané des n opérations les plus coûteuses en temps et/ou les plus fréquentes en cours d'exécution telles que l'écriture du cache ou l'exécution de formules.

Par défaut, la commande « LIRE APERCU ACTIVITE » traite des opérations effectuées en local (avec 4D monoposte, 4D Server ou 4D en mode distant). Avec 4D en mode distant cependant, vous pouvez également obtenir l'aperçu des opérations effectuées sur le serveur : il suffit pour cela de passer l'étoile (*) en dernier paramètre. Dans ce cas, les données du serveur seront récupérées localement. Le paramètre * est ignoré lorsque la commande est exécutée sur 4D Server ou 4D monoposte.

Une nouvelle commande est également apparue en version 4D v16 R4 (modifiée en v16 R5) : « Lire activite process ». Elle retourne une vue instantanée des sessions des utilisateurs connectés et/ou des process exécutés à un instant précis, y compris les process internes qui n'étaient pas accessibles avec la commande « INFORMATIONS PROCESS ».