PRÉCONISATIONS POUR VOS APPLICATIONS 4D

Dernière mise à jour : 2 février 2017

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
- Utiliser une structure compilée

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 (http://doc.4d.com/4Dv15/4D-Internet-Commands/15/Annexe-B-Numeros-des-ports-TCP.300-2397835.fe.html)

IV- Processeur

Recommandé :

- plusieurs 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.

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 matrices de certification sont disponibles sur notre site web : http://www.4d.com/fr/support/resources.html.

- Sous Windows :
    - A partir de la version 4D v15,
        - pour 4D : opter pour **Windows 10**,
        - pour 4D Server ou pour vos serveurs TSE : opter pour **Windows Server 2008 R2** (la gamme 4D v14 et supérieures sont incompatibles avec Windows Server 2008 Standard) ou pour **Windows Server 2012 R2** (les versions 4D Windows ne sont pas compatibles avec l'option « Server Core » de Windows Server) ;

    - Déployer 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 Developer est disponible en version 64 bits à partir de la version v15 R5 ;

    - 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.

- Sous Mac :
    - Déployer la gamme 64 bits :
        - 4D Server est disponible en version 64 bits à partir de la version v15.1,
        - 4D Developer est disponible en version 64 bits à partir de la version v15 R5.

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 de nouvelles fonctionnalités puissantes comme la possibilité de gérer les process multithread, les impressions entièrement basées sur le système ou les nouveaux éditeurs d'états rapides et d'étiquettes.

VI- Mémoire

Recommandé :
- 8 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

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 les sauvegardes

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

Recommandé :

- Utiliser le mécanisme des sauvegardes de 4D
- Activer le fichier d'historique
- Décocher l'option « Annuler l'opération au bout de X tentatives » dans l'onglet « Sauvegarde » des propriétés de votre base 4D
- 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 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 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 v16

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)

- utiliser une base compilée

- 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 virtualisé 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 !