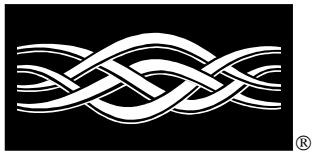
Microsoft Windows NT 4.0 Option Pack

Microsoft® Internet Information Server



Livre Blanc

Internet Information Server : Optimiser son serveur Web

Lionel CAU, Mars 2000.

Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • USA

Microsoft®
Internet Information Server

Abstract

Ce document vous permettra de découvrir comment optimiser au mieux votre serveur web IIS. Vous y découvrirez les paramètres systèmes et application à tester dans votre environnement ainsi que la démarche à adopter pour ajuster au mieux ces paramètres. Les informations regroupées dans ce document proviennent de différents livres blancs en langue anglaise, du ressource Kit et de conseils standards concernant le développement d'applications sous IIS. Il s'adresse avant tout aux administrateurs et, pour la partie 6 seulement, aux développeurs.

SOMMAIRE

Introduction 6

1. Rappels sur IIS 7
1.1. Qu'est-ce qu'IIS ? 7
1.2. Comment administrer IIS ? 8
1.3. Quelles sont les données fournies par IIS ? 9
1.4. Que faut-il optimiser alors ? 10

2. La mémoire 11
2.1. Les besoins 11
2.2. La mémoire pour NT 11
2.3. La mémoire pour IIS 12
2.4. Conclusion: Conseils d'optimisation concernant la mémoire 13

3. Les processeurs 16
3.1. Les besoins 16
3.2. Les processeurs pour NT 17
a) La ressource utilisée 17
b) Contrôle des interruptions et des DPC 17
c) DPCs et cartes réseaux 18
3.3. Les processeurs pour IIS 19
a) Les connexions HTTP persistantes 19
b) Le nombre de connexions 20
c) Parlons threads 21
3.4. Conclusion: Conseils d'optimisation concernant les processeurs 22

4. Le réseau 24
4.1. Les besoins 24
4.2. Le réseau pour NT 24
a) La bande passante : 24
b) La capacité du réseau 25
4.3. Le réseau pour IIS 26
a) Mesurer les performances 26
b) Le contrôle de la bande passante 26
4.4. Conclusion: Conseils d'optimisation concernant le réseau 28

5. La Sécurité 29
5.1. Les besoins 29
5.2. La sécurité pour NT 29
5.3. La sécurité pour IIS 30
a) Stressez votre site 31
b) Quels compteurs examiner ? 31
c) Au niveau des composants compilés 32
5.4. Conclusion: Conseils d'optimisation concernant la sécurité 32

6. Les Applications 34
6.1. Les compteurs Perfmon 35
6.2. WAM 35
6.3. Les principaux ajustements 37
a) Ajustement de la réserve de threads et de la file d'attente ASP 37
b) Augmentez le nombre de moteurs de scripts mis en cache 38
c) Ajustez le nombre de fichiers ASP mis en cache 40
d) Ajustez le 'timeout' des scripts ASP 41
e) Paramétrage et metabase 41
6.4. Les scripts côtés client 42
6.5. Les moteurs de script 42
6.6. Gérer au mieux son cache 43
6.7. Optimisez votre code ASP 45
6.8. Les composants COM 45
6.9. Les variables sessions et applications 46
6.10. Les accès aux bases de données 47

7. IIS 5.0 et les gains en performance 49
7.1. Nouveautés et améliorations concernant le service WWW 49
7.2. Nouveautés et améliorations concernant ASP 51

Conclusion 53

Bibliographie 54

Introduction

Dans le cadre de ce document, il faut considérer avant tout Internet Information Server (IIS) comme un serveur de protocole. Le code des pages qui tourne dans un serveur Internet étant le fruit d'un développement personnel, l'optimisation d'un serveur IIS est en très grande partie une question d'optimisation de ce code et non une question d'ajustement de paramètres propres au comportement d'IIS en tant que service NT.

Certes, et nous le verrons dans la majeure partie de ce document, ces paramètres existent et peuvent améliorer les performances par rapport à votre environnement et votre utilisation d'IIS mais gardez à l'esprit que si un code est mal écrit, si il entraîne une lenteur d'exécution, vous ne pourrez jamais contrebalancer cette lenteur par une optimisation du service Web.

Dans une première partie, nous ferons des rappels très succincts concernant IIS. Puis nous adopterons pour chaque chapitre suivant: la mémoire, les processeurs, le réseau et la sécurité un plan identique a savoir:

- quels sont les besoins ?
- quels sont les concepts à connaître pour optimiser la ressource au niveau du système d'exploitation (à savoir NT) ?
- quels sont les concepts à connaître pour optimiser la ressource au niveau d'IIS lui-même ?
- et pour finir un résumé sur les conseils d'optimisation.

Le chapitre 6 sera dédié aux applications
Le chapitre 7 donnera un aperçu des nouvelles fonctionnalités d'IIS 5.0 relatives aux performances.

1. Rappels sur IIS

1.1. Qu'est-ce qu'IIS ?

IIS est un serveur de protocoles. C'est plus précisément un jeu de services utilisant les protocoles Internet les plus répandus :
- HTTP
- FTP
- NNTP (Network News Transfer Protocol)
- SMTP (Simple Mail Transfer Protocol)
- HTTP: le plus célèbre pour servir des pages au format HTML

Note importante: Ce document ne traitera que de l'optimisation du serveur HTTP. A ce titre, par abus de langage, nous désignerons dans la suite de ce document par "IIS" ou "serveur Web" le serveur HTTP uniquement.

D'autre part, le produit s'accompagne d'un jeu d 'APIs pour étendre les possibilités du serveur :
- CGI (Common Gateway Interface)
- ISAPI (Internet Server API)

IIS est en fait le centre Internet de votre serveur NT, c'est le moteur sur lequel vont s'appuyer l'ensemble des logiciels apparentés à Internet. Les fonctionnalités qu'il propose étant:

- Etablir et maintenir des connections HTTP
- Lire les requêtes HHTP et écrire les réponses
- Modifier les entêtes HTTP
- Obtenir les informations des certificats clients
- Gérer les connexions asynchrones
- Mapper les URLs sur des chemins physiques
- Gérer et faire tourner les applications
- Transmettre les fichiers

L'image suivante reflète cette position dans le cadre de la panoplie des outils Microsoft:

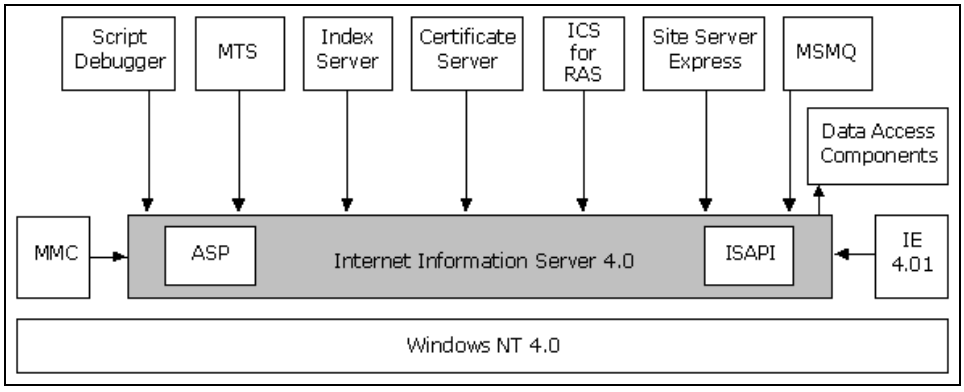


Figure 1 : IIS, le centre internet de votre serveur NT

1.2. Comment administrer IIS ?

Pour administrer son serveur IIS, les outils sont:
La Microsoft Management Console (MMC) qui est l'outil à employer de préférence et qui donne une unique interface pour gérer tous les aspects du service.
Par la MMC, vous modifiez des propriétés et des valeurs qui sont enregistrées dans la metabase. Pour se représenter la metabase, vous pouvez faire le parallèle suivant: la metabase est à IIS ce que la base de registre est à NT.
En effet, toutes les informations propres à la configuration d'IIS sont stockées dans ce fichier dédié qu'il est possible de modifier également:

- par programmation (comme un script via ADSI)
- par des utilitaires en ligne de commande tels que ceux fournis en standard avec le produit (scripts d'administration et ADSUTIL.vbs se trouvant dans le répertoire %systemroot%\system32\inetsrv\adminsamples)
- par des utilitaires graphiques tels que MetaEdit

Une autre façon de modifier la metabase est d'utiliser l'administration par HTML.

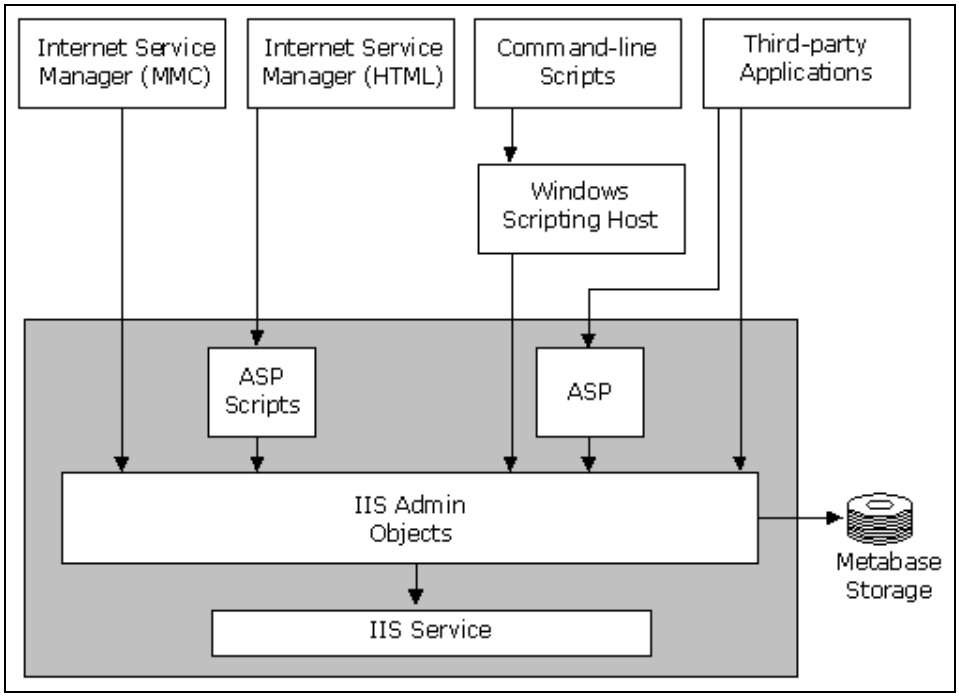


Figure 2 : Les différents moyens de paramétrage et d'administration d'IIS

D'un autre côté IIS est également un service NT, à ce titre il est également possible de modifier des paramètres spécifique au service Web dans la base de registre du système.

Dans la suite de ce document, gardez à l'esprit que tout ajustement dans la MMC pourra être effectué en modifiant directement dans la metabase. Certains autres paramètres ne pourront se faire qu'en modifiant la base de registre.
Dans tous les cas, un redémarrage du service Web sera nécessaire pour prendre en compte les modifications.

Pour arrêtez les services Web entrez la commande suivante dans une console DOS:

NET STOP IISADMIN /Y

Attention: Ceci stoppera également tous les services qui s'appuient sur iisadmin (services FTP et SMTP, la majorité des services de Site Server, Proxy server, etc…)

Pour redémarrer le service Web, lancez la commande:

NET START W3SVC

Remarque: cela ne relancera pas les autres services internet pour autant, vous devrez les redémarrez un à un.

D'autre part, Vous pourrez également être amené à modifier des paramètres concernant NT lui-même, toujours via la base de registre ou par l'interface générale. Dans ce cas le redémarrage du service Web sera sans effets et vous devrez redémarrer la machine en totalité.

1.3. Quelles sont les données fournies par IIS ?

IIS fournira par HHTP les pages que vous avez écrites. Les applications web executeront votre code ASP, lanceront les composants que vous avez mentionné (voir même créé). Optimiser son serveur c'est avant tout optimiser le code de vos applications.

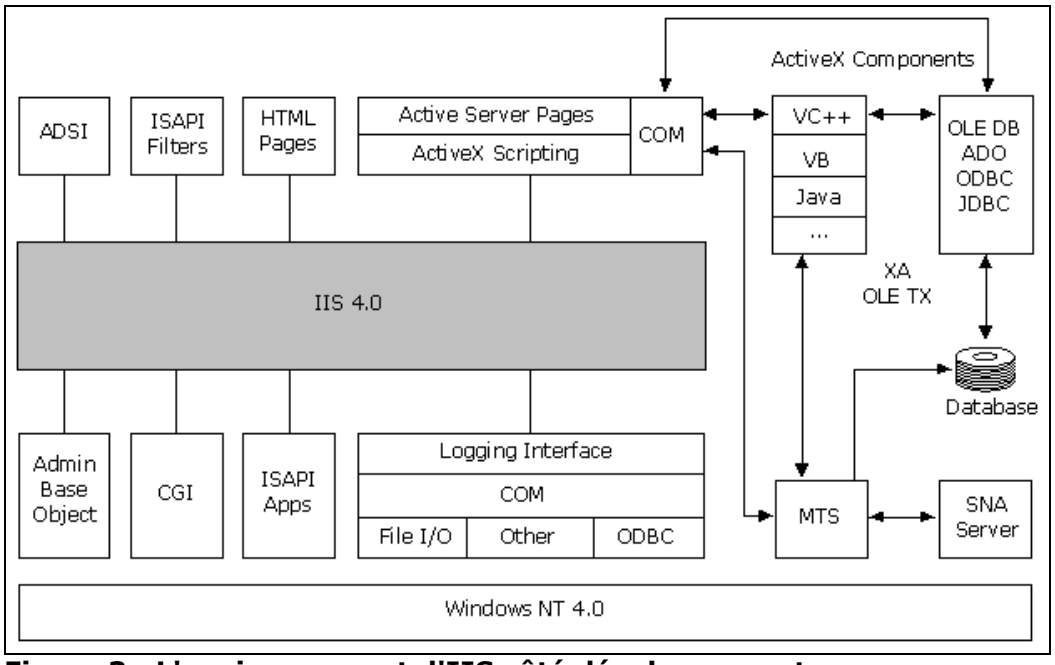


Figure 3 : L'environnement d'IIS côté développement

1.4. Que faut-il optimiser alors ?

L'optimisation d'un serveur IIS passe dans un premier temps, nous l'avons déjà dit, par l'optimisation des applications. Mais il est certain que si le système sur lequel s'appuie le service Web est sous dimensionné par rapport à la charge demandée, les performances générales s'en trouveront diminuées.
Pour vérifier que le système est correctement dimensionné et configuré il faut analyser les besoins et les disponibilités concernant les ressources standards:

- la mémoire
- les processeurs
- le réseau

En conséquence, le schéma suivant représente les prochains chapitres que nous verrons dans ce document.

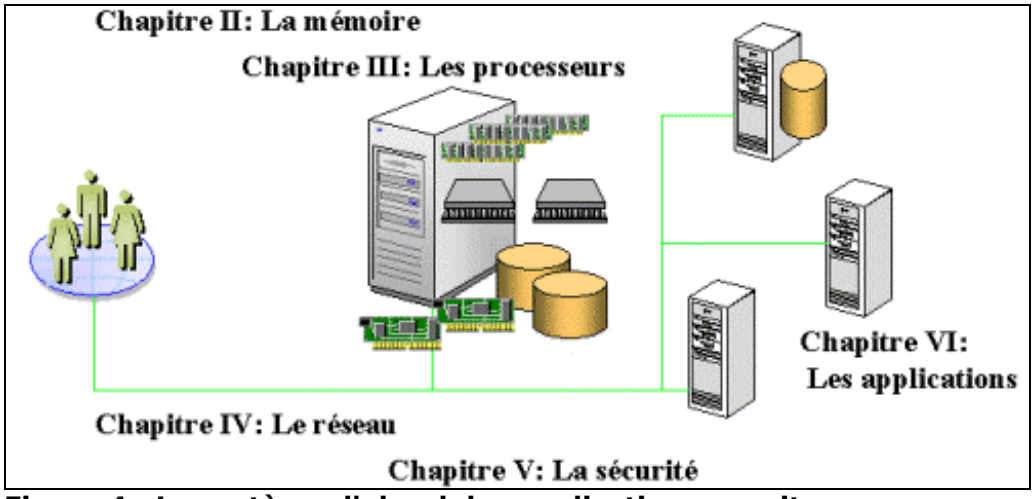
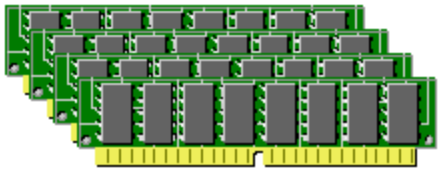


Figure 4 : Le système d'abord, les applications ensuite

Note importante: De tous les paramètres que nous verrons par la suite, il est important de savoir qu'**il n'existe pas de valeurs de références**. Tout ajustement doit être considéré par rapport à l'environnement du serveur, à ce qui lui est demandé, à ce quoi il est destiné. Il est évident qu'un serveur intranet ne devra pas être ajusté de la même manière qu'un serveur internet. De même un serveur ou les pages ASP sont rares et ne délivrant que des pages statiques ne répondra pas au mêmes besoins qu'un serveur d'applications internet.

Vous devrez avant tout comprendre la définition d'un paramétrage, modifier en conséquence votre serveur puis en observer les conséquences avant de valider cet ajustement. Ensuite, et seulement ensuite, vous pourrez passer à un autre paramètre.

2. La mémoire



2.1. Les besoins

Dire qu'IIS et NT ont besoin de mémoire vive pour fonctionner est un évidence. Dire que plus la quantité de RAM présente est grande plus cette ressource ne sera pas limitative vis à vis des performances en est une autre.

Par contre, il est important de savoir quels sont les besoins en mémoire d'IIS et comment ces besoins sont répartis.

Qu'est-ce qui est stocké dans la mémoire par un serveur IIS ?

- IIS lui-même: 2.5 Mo, cela correspond à l'espace de travail d'inetinfo (2 Mo + 0,5 Mo pour ASP)
- Les fichiers des pages web fréquemment utilisés par IIS: le nombre et la taille varient suivant l'installation. Ils sont stockés dans le cache du système de fichiers, un emplacement de la mémoire physique réservé aux pages fréquemment employées.
- Les objets fréquemment utilisés: objets fréquemment utilisés par le service IIS (handles de fichiers, listes de répertoires, etc…)
- Le fichier du journal d'IIS (par blocs de 64 Ko)
- Le Tableau TCB (TCP tient a jour une table de blocs de contrôle de transmission afin de stocker des données pour chaque connexion TCP, IIS a besoin de cette table et va augmenter sa taille). Ceci est de la mémoire non-paginable.
- Le réserve de threads, non-paginable également.

2.2. La mémoire pour NT

Il convient de s'assurer que la quantité de mémoire disponible est suffisante à NT avant même de se demander si ce que va fournir NT à IIS est suffisant.

Pour déterminer si une ressource est suffisante et comme dans tout problème d'optimisation sous NT, l'outil à employer est l'analyseur de performance (Perfmon).

Pour cela gardez à l'esprit les points suivants:
- il faut observer l'activité pendant une période d'activité normale (et non un creux comme la nuit) et prévoir toujours les accroissements occasionnels (pointes le matin à l'arrivée des utilisateurs sur site par exemple)
- il faut observer l'activité sur une durée suffisamment grande pour être parlante (un log pris sur dix minutes ne sera pas représentatif)

Sachez de plus, que le système de mémoire virtuelle de NT est conçu pour ajuster de manière automatique ses capacités, vous ne pourrez pas fixer des quantités de ressources à une valeur bien précise. Ce que vous pourrez faire consistera à indiquer des valeurs de départ, des valeurs minimum ou maximum mais pas des valeurs exactes. Le principal objectif du contrôle de la mémoire d'un serveur NT exécutant IIS consiste donc à s'assurer que le serveur dispose d'assez de mémoire physique et

non à ajuster la taille de chaque composant comme cela pourrait être le cas avec d'autres systèmes d'exploitation.

De même IIS s'auto-ajuste. Il règle par exemple la taille du cache de ses objets, la remarque précédente est valable pour IIS également.

Les compteur perfmon a analyser sont:
- Objet "Memory", compteur "available bytes": devrait être toujours au dessus de 4 Mo. Si il est en dessous, cela impacte sur la taille du working set alloué aux processus et donc sur la pagination.

- Objet "Paging File", les deux compteurs ("usage" et "usage peak"): une trop forte utilisation du fichier de pagination indique un goulot d'étranglement au niveau de la mémoire.

- Les deux compteurs de l'objet "Memory": "Page Faults/sec" et "Cache Faults/sec" sont également intéressants pour voir les fautes de pages dans le working set des processus ou du cache.

- Les fautes de page 'Hard' sont les plus pénalisantes car elles entraînent une lecture sur le disque. Le ratio fautes de pages 'Hard' / fautes de pages en général indique un sous-dimensionnement mémoire. Pour mesurer les fautes de pages "Hard", il faut prendre les compteurs "Page Reads/sec" de l'objet "Memory". C'est le compteur "Page Faults/sec" vu dans le point précédent qui servira pour quantifier les fautes de pages en général.

2.3. La mémoire pour IIS

Vous connaissez la quantité de mémoire totale embarquée sur votre système, une fois que vous avez analysé l'utilisation qu'en fait NT, regardez ce qu'il en reste pour le fonctionnement d'IIS.

Définition: L'espace de travail (ou "working set") d'un processus est la quantité de mémoire visible par un processus.

L'espace de travail d'inetinfo (le processus d'IIS) est partagé par les services ASP, WWW, FTP et SMTP. Vous devrez donc vous attachez à la taille de ce working set et vous assurer que celle-ci permet à ces services de tourner au mieux. Il n'y a pas de taille standard pour le "working set' d'inetinfo (voir remarque en fin du chapitre 1)

Moins une application pagine, plus ses performances seront bonnes. Pour que IIS fonctionne efficacement, la taille de son Working set doit donc lui permettre d'y maintenir son code en totalité, les fichiers utilisés le plus fréquemment, le cache IIS, les buffers de données pour le logging, et ses structures propres.

Les compteur perfmon a analyser sont:

- Objet "Memory", compteur "available bytes": le système réduit la taille des working set si ce nombre devient inférieur à 4 Mo.

- Objet "Process", compteur "working set" sur le process "inetinfo": pour la taille du working set
- Objet "Process", compteur "Page Fault/sec" sur le process "inetinfo": pour les fautes de pages "Hard" ET "soft" de inetinfo
- Objet "Memory", compteur "Page Faults/sec": pour les fautes de pages "Hard" et "Soft" sur tous les working sets
- Objet "Memory", compteur "Page Reads/sec": pour les fautes de pages "Hard" (une valeur de 5 indique un sous-dimensionnement mémoire)
- Objet "Memory", compteur "Pages Input/sec": nombre de pages lues pour compenser les fautes de page. C'est un indicateur du 'coût' des fautes de pages.

IIS dispose d'un cache nommé le "*IIS Object Cache Data*", ce dernier fait partie du working set d'inetinfo. Ce cache peut être paginé sur le disque si la mémoire n'est pas suffisante pour supporter un working set suffisamment large pour inetinfo lui-même. Il est important d'avoir assez de mémoire pour maintenir le cache dans le working set. Augmentez le contenu de la valeur *ObjectCacheTTL* sous *HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\InetInfo\Parameters* si vous suspectez que le cache est rafraîchit trop souvent au regard des compteurs de l'objet "Internet Information Services Global":
- Cache Hits
- Cache Misses
- Cache Hits %
- Cache Flushes
- Objects

N'oubliez pas également que comme tout service demandant un fichier, IIS s'appuie sur le cache du système de fichiers.
Exemple: Supposons qu'un thread d'IIS doit ouvrir un fichier. Il faut d'abord stocker le handle sur ce fichier dans le cache d'objet d'IIS puis stocker les données du fichier dans le cache du système de fichiers.

C'est donc l'objet "Cache" qui est à analyser:
- "MDL Reads/sec"
- "MDL Read Hits %" (l'inverse est donc équivalent au compteur "Cache Faults" de l'objet "Memory")
- "Read Aheads/sec" (basé sur l'algorithme de prédiction du gestionnaire de cache : de plus larges blocs de données sont lus à chaque opération de lecture lorsqu'une lecture séquentielle est détectée)

Définition: MDL = Memory Descriptor List. Les lectures MDL sont des opérations de lecture dans lesquelles le système utilise une liste d'adresses physiques pour chaque page afin d'en simplifier la lecture. Les lectures MDL servent souvent à récupérer des fichiers de pages Web et FTP mis en cache.

2.4. Conclusion: Conseils d'optimisation concernant la mémoire

- Améliorez la distribution des données: procédez à une défragmentation régulière du disque et surtout conservez les fichiers Web liés entre eux sur les partitions logiques d'un même disque:

Si le taux de défauts de cache est élevé, rajoutez de la mémoire ou pensez à une redistribution efficace des données sur le disque. Des fichiers stockés sur les mêmes partitions logiques ont plus de chance de tirer parti des stratégies d'optimisation du gestionnaire de cache de Windows NT.

- Utilisez NTFS
- Eliminez / Arrêtez les services inutiles
- Recourrez à la mise en miroir et aux agrégats par bande
- Remplacez ou convertissez les applications CGI
- Accroissez les fichiers d'échange (ajoutez-en, agrandissez-les)
- Modifiez les valeurs temporelles du cache d'objets IIS (ObjectCacheTTL)
- Modifiez l'équilibre entre cache du système de fichiers et espace de travail IIS. Par défaut, lors de l'allocation de l'espace de mémoire, les serveurs exécutant Windows NT sont configurés pour donner la préférence au cache du système de fichiers par rapport aux espaces de travail des processus. Bien que les serveurs IIS bénéficient d'un large cache de système de fichiers, ce réglage entraîne souvent l'écriture sur disque de tout le code paginable d'IIS ce qui ralentit son traitement. Afin de l'éviter, réglez les propriétés du service serveur (dans les propriétés du réseau) sur l'option "Débit maximal pour les applications réseau" (redémarrage nécessaire).

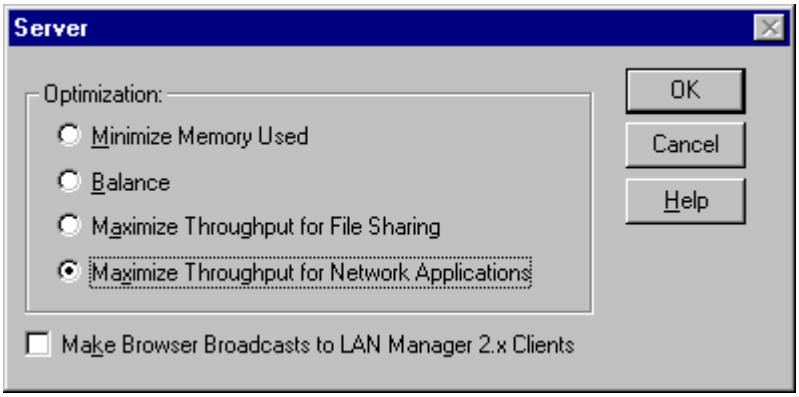


Figure 5 : Favoriser un "Débit maximal pour les applications réseau"

- Limitez les connexions. Si le serveur manque de mémoire la limitation du nombre de connexions peut améliorer la situation (10Ko pour chaque connexion afin de gérer les structures de données qui en assurent le suivi). FTP est limité à 1000 par défaut mais pas le Web (ni le SMTP d'ailleurs)

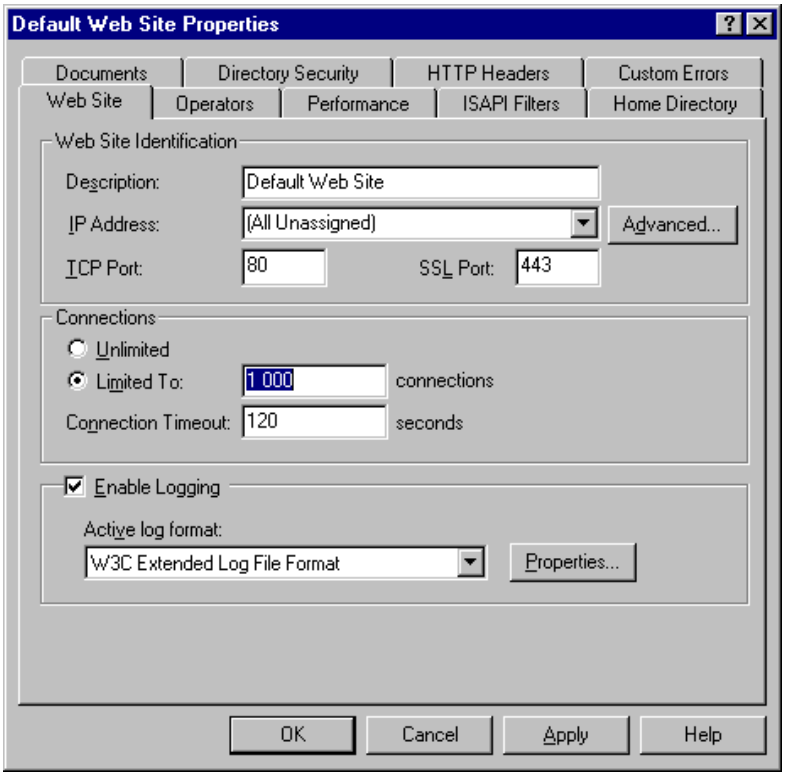
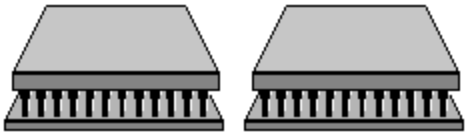


Figure 6 : Limitez les connexions si cela est envisageable

3. Les processeurs



3.1. Les besoins

La même remarque faite concernant les besoins en mémoire est valable pour la ressource processeur, mais il faut savoir que un trop grand nombre de processeurs peut nuire aux performances d'un serveur. Comme toujours, il faut évaluer les besoins et y pourvoir mais ne pas tomber dans l'excès.

Le code d'IIS est multi-thread, dans le cas d'un goulet d'étranglement de la ressource processeur, les threads actifs du processus doivent attendre dans une file d'attente que des ressources se libèrent. Dans ces conditions, rajouter de la mémoire, des connexions réseau ou des disques ne servira à rien.

Dans ce chapitre nous verrons qu'il est possible d'ajuster le nombre de threads et répartir ces derniers sur différents processeurs.

Note: Quand vous analysez la charge processeur, ce ne sont pas les pics d'activité qui doivent capter votre attention. Un processeur qui est à 100% d'utilisation occasionnellement est beaucoup moins préoccupant qu'un processeur étant constamment entre 70% et 80% d'utilisation.

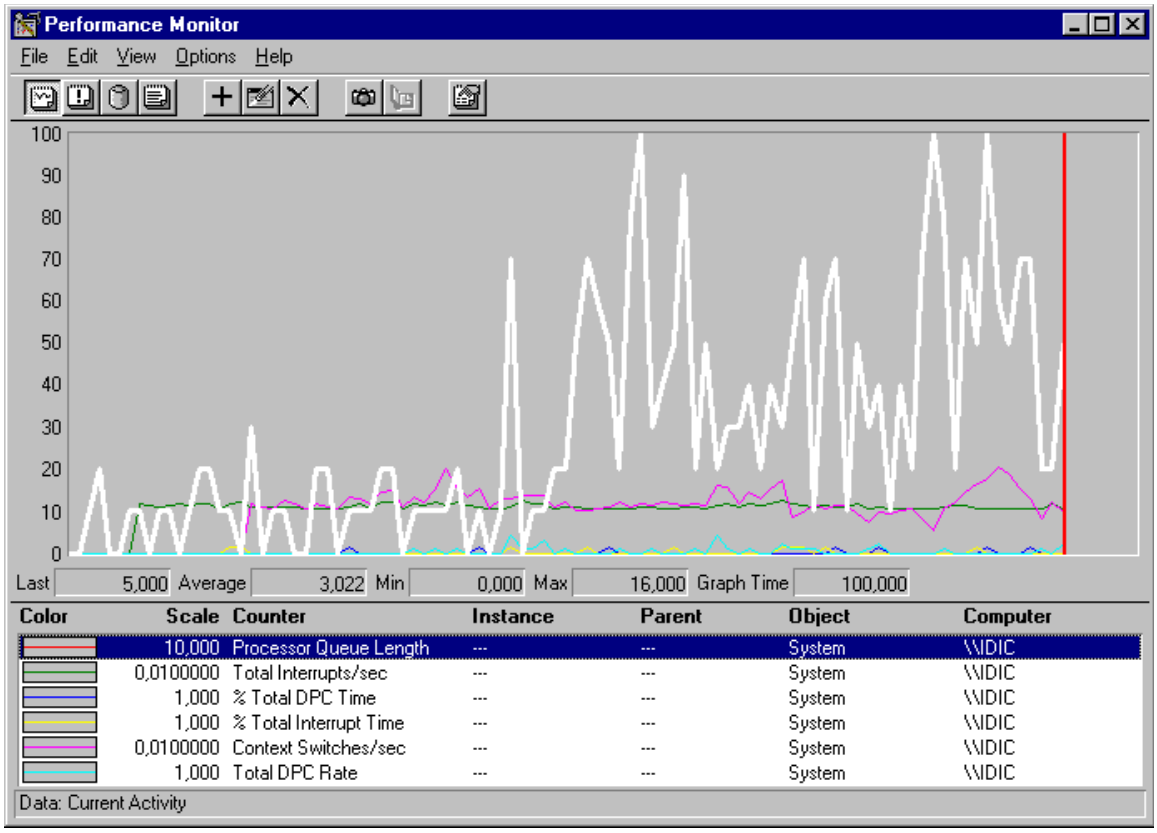


Figure 7 : Un exemple d'une ressource processeur ayant faire à un pic d'activité mais dont le dimensionnement permet d'assurer la charge

3.2. Les processeurs pour NT

a) La ressource utilisée

Il convient de s'assurer que la ressource CPU disponible est suffisante à NT avant même de se demander si ce dont va pouvoir disposer IIS est suffisant.

Le ou les processeurs d'un serveur doivent supporter le système d'exploitation et tous les processus relatifs aux services Internet.

Les compteurs perfmon à analyser sont:

- Objet "System", compteur "Processor Queue Length": nombre de threads en attente. Une valeur de 2 en continu indique un problème processeur.
- Objet "System", compteur "% Total Processor time": si trop important cela signifie une charge significative des processeurs
- Objet "Processor", compteur "% Processor time": si très différent du précédent cela indique un problème de distribution de la charge entre les processeurs.
- Objet "Processor", compteurs "% Privileged Time" et "% User Time": à comparer pour voir si le système travaille plus pour NT lui-même ou pour IIS (et les autres processus en mode user)

Remarque: Nous parlons dans ce paragraphe de NT et non d'IIS donc vous noterez que les compteurs ci-dessus n'ont aucun rapport avec le service web mais sont côté système.

La cause la plus courante d'un apparent goulet d'étranglement du processeur provient de la mémoire. Si le système ne dispose pas d'assez de mémoire physique pour stocker le code et les données nécessaires aux programmes, le processeur perdra une partie notable de son temps en opérations de pagination.

b) Contrôle des interruptions et des DPC

Les interruptions sont des signaux à très haute priorité qui suspendent l'activité du processeur et le préparent à une nouvelle activité ne fut ce que pour un temps très court. Elles consomment du temps processeur en interrompant le travail mais elles sont essentielles a un système multitâche préemptif.

Si la ressource processeur d'un système est continuellement interrompue de la sorte par quantités d'interruptions, les performances générales vont s'en trouver dégradées avec un impact certain sur un éventuel service web présent sur la machine. Dans le cadre d'un serveur IIS dont l'activité principale est basée sur le réseau les interruptions à surveiller sont plus particulièrement celles générées par les cartes réseau.

Définition: DPC = appels de processus différés (Deferred Procedure calls). Ce sont comme des interruptions mais elles peuvent être retardés pour permettre au

processeur de mener à bien une tâche de plus haute priorité (IRQL inférieur aux interruptions).

L'adaptateur réseau produit une DPC lorsqu'il reçoit un nouveau paquet ou complète une transmission.
Le système collecte dans ce cas quelques informations de base puis ajoute un ou plusieurs DPC à la file afin de gérer les étapes suivantes du processus. Lorsque le DPC reçoit du temps processeur, le travail de mise en place de la connexion peut s'enclencher

Les serveurs Pentium/Risc sont à même de contrôler des milliers d'interruptions et DPCs sans s'absorber complètement dans cette tâche . Un serveur actif IIS sur une bande passante importante risque toutefois d'interrompre le processeur trop souvent. Pour améliorer cela, quelques-uns des plus récents pilotes de carte réseau disposent d'une fonctionnalité avancée appelée modération des interruptions . Lorsque le pilote détecte un taux d'interruptions élevé issu de l'adaptateur il déclenche une désactivation des interruptions et les accumuler dans un tampon. Quand le processeur a fini son travail la réactivation des interruptions a lieu.

Sur un système monoprocesseur, trop d'interruptions montre en général soit qu'une carte réseau ou un disque pose problème soit un processeur surchargé. Sur un multiprocesseurs cela peut également montrer une répartition médiocre entre les processeurs de la charge de travail.

Les compteurs perfmon à analyser sont:
- Objet "Processor", compteur "% Interrupt Time": le temps passé à gérer ces interruptions. Si "% Processor time" est supérieur à 90% et cette valeur est supérieure à 15% alors le processeur est probablement étouffé par les interruptions. Un goulot d'étranglement processeur est alors suspecté.
- Objet "Processor", compteur "% DPC Time": de même mais pour les DPCs.

Si un problème de répartition entre plusieurs éventuels processeurs est détecté, la première mesure à prendre est alors de s'assurer de la mise en place d'une HAL adéquate et à jour.
Comparez "Processor", compteur "% Interrupt Time" et "Processor", compteur "% DPC Time" avec celles du compteur "% Processor Time".

Si un processeur passe le plus clair de son temps a satisfaire des interruptions et des DPCs: mettez à jour vos processeur(s) ou rajoutez-en.

Par contre si "% Processor Time" est acceptable (inférieur à 85) et si "Processor", compteur "% Interrupt Time" et "Processor", compteur "% DPC Time" sont valables (inférieur à 20) ce n'est donc pas un problème.

c) DPCs et cartes réseaux

La distribution des DPCs produits par les cartes réseau est une fonction de ndis.sys, (l'implémentation Windows NT de la spécification d'interface des pilotes réseau qui cache les détails de la carte au reste du système d'exploitation). Par défaut, il associe à chaque carte réseau un processeur. En fait, il tente de compenser le fardeau

qu'encourt le processeur de plus bas numéro en associant la première carte réseau au processeur de plus haut numéro. Chaque carte suivante est associée au processeur subséquent, dans l'ordre décroissant des numéros des processeurs.

Sur de nombreux systèmes, cette stratégie aboutit a une répartition effective de la charge de travail. Si un serveur qui ne distribue pas ses interruptions dispose de deux processeurs et d'une carte réseau, le processeur 0 satisfera les interruptions et le processeur 1 les DPCs.

Malheureusement, il y a des DPCs qui sont générés par des interruptions. Ceci implique un basculement générant lui même une interruption inter-processeur (très haute priorité, temps consumé), plus des données à vider du cache du processeur expéditeur vers le destinataire qui doit collecter tout ca… donc un coût non négligeable.

Heureusement, il est possible d'avoir une amélioration des performances en optimisant la distribution des DPCs: Cela dépend si le système distribue ou non les interruptions.

- Si aucune distribution n'est faite par le système lui-même et si le serveur est multiprocesseurs et si le processeur de plus haut numéro fonctionne a 100% et que plus de la moitié de son temps est occupé a faire du temps DPC (>50) alors :
  - adoptez un système qui distribue les interruptions
  - ajoutez des cartes réseau pour en avoir une par processeur (ndis.sys)… et ajoutez donc de la bande passante.

- Si une distribution symétrique est gérée par le système lui-même (distribution statique ou dynamique) et si le serveur est multiprocesseurs et alors vous pouvez régler *ProcessorAffinityMax* à 0. Dans ce cas, les cartes réseau ne sont pas associées aux processeurs (vous désactivez cette fonctionnalité de ndis.sys) et les DPCs demeurent sur le processeur qui a contrôlé l'interruption. Cette valeur se trouve dans *HKEY_LOCAL_MACHINE\system\CurrentControlSet\Services\NDIS*

3.3. Les processeurs pour IIS

a) Les connexions HTTP persistantes

Chaque connexion qu'établit un service IIS consomme un fragment du temps processeur. La carte réseau interrompt le processeur pour lui signaler l'arrivée d'une requête.
Donc, pour chaque requête il faut: modifier la table TCB (priorité de la connexion, numéro de sockets locaux et distants), satisfaire la requête, et quand la connexion se referme supprimer les structures qui l'ont prise en charge. Tout ceci a un bien évidemment un coût processeur.

Les connexions HTTP persistantes est un moyen de diminuer le coût processeur d'IIS en matière de connexions: ce la consiste à conserver une connexion même après que la demande initiale de connexion soit complétée et ainsi pouvoir la réutiliser.

Note: Le client et le serveur doivent tous deux prendre en charge cette fonctionnalité (IIS 1.0 + et IE 2.0 + et Netscape 2.0 +).

Cette fonctionnalité est activée par défaut.

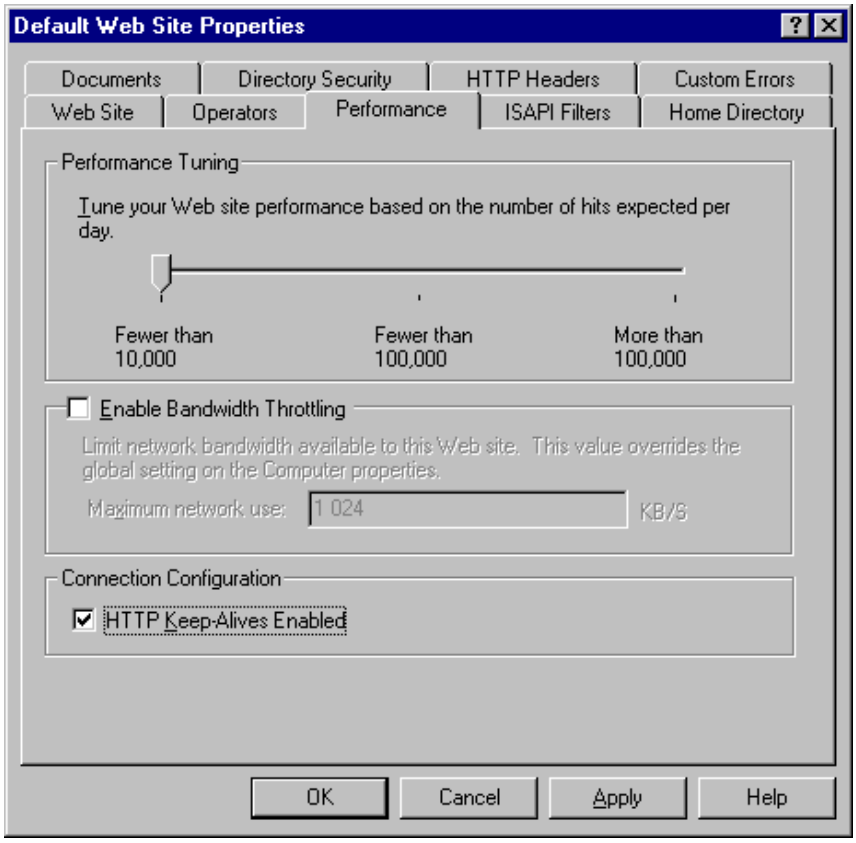


Figure 8 : Vérifiez que les connexions HTTP persistantes sont activées

Attention: Les connexions HTTP persistantes sont différentes de connexions TCP/IP persistantes (cette technologie étant basée sur des messages envoyés régulièrement pour déterminer si une connexion est toujours active)

Remarque: Pour mesurer l'impact de la prise en charge ou non des connexions HTTP persistantes, le plus simple et le plus parlant consiste à tester via perfmon les performances avec et les re-tester sans !

b) Le nombre de connexions

Nous avons donc vu qu'une connexion coûte en terme de charge processeur. Il convient donc d'analyser leur nombre et l'évolution de ce nombre.
Plus de connexions sont à prendre en charge, plus la consommation CPU sera importante.

Au niveau de l'analyseur de performances, vous pouvez contrôler le nombre de connexions simultanées aux services IIS an regardant dans l'objet "Web Service" les compteurs "Current Connections" et "Maximum connections".

Remarque: Ces compteurs affichant la dernière valeur et non une moyenne, ils sont à relever sur une durée probatoire pour évaluer cette moyenne.

Attention: Ces compteurs sont au niveau de la couche Application OSI. En conséquence, le décompte des connexions TCP/IP risque de ne pas égaler les connexions HTTP, FTP et STMP si les connexions ont été bloquées, rejetées ou ré-initialisées entre les couches Transport et Application.

Il vous appartient donc de vérifier que le serveur soit dimensionné de manière à ce que le nombre de connexions à gérer soit compatible avec les ressources CPU disponibles.

c) Parlons threads

Définition: Un thread est une séquence d'exécution de chaque processus qui exécutent le code du processus sur le processeur.

Le rapport entre les threads, les connexions et les demandes est complexe du fait qu'IIS se sert du modèle de threads travailleurs plutôt que du modèle plus simple, mais moins efficace, d'un thread par client. Dans ces conditions, un thread va prendre en charge l'établissement et la gestion d'une connexion alors que d'autres threads vont servir à l'identification, l'analyse des demandes, la localisation des fichiers , l'ouverture et la transmissions de ces fichiers, le maintient des structures internes à IIS, etc…

Donc il n'est pas possible d'associer des threads individuels à des connexions:
Il n'existe aucune association simple entre le nombre de threads et le nombre de connexions, entre threads et demandes, entre le nombre optimal de threads dans le processus et le nombre de fichiers servis ou encore le nombre de demandes satisfaites ou le nombre de connexions conservées.

Par contre, il est possible de mesurer le nombre de threads et de regarder la portion de temps processeur qu'ils consomment et ainsi comparer l'évolution de leur nombre suivant le nombre de demandes et mesurer l'impact sur l'utilisation du ou des processeurs.

Il est donc possible de mesurer avec l'analyseur de performance le nombre de threads dans le processus IIS, la portion de temps processeur que reçoit chaque thread et associer le nombre de threads (et l'activité du processeur) au nombre de connexions en cours, au nombre de fichiers servis, etc…

Optimisation des valeurs des threads:

Par défaut, il y a un maximum de dix threads par processeur. Cette valeur est en ajustement constant du nombre de threads par rapport à l'activité.

Si presque tous les threads du processus IIS sont occupés la plupart du temps et que les processeurs sont à leur capacité maximum ou l'avoisinent, il faudra:

- répartir la charge sur plusieurs serveurs.
- ou ajouter des processeurs (attention: des processeurs inutiles ou sous-employés entraînent des dégradations de performances)

Si par contre les threads semblent occupés mais pas les processeurs, vous pouvez envisager l'augmentation du nombre de threads **dans des limites acceptables**

Pour cela, la valeur "Maxpoolthreads" est à rajouter dans la base de registre sous *HKEY_LOCAL_MACHINE\System\CurrentcontrolSet\Services\Inetinfo\Parameters*
Le conseil classique consiste à ne pas affecter une valeur inférieure à 5 ou supérieure à 20 à ce paramètre.

3.4. Conclusion: Conseils d'optimisation concernant les processeurs

- Vérifiez la mémoire avant de vous pencher sur les processeurs

- Si la plupart des threads d'IIS sont occupés tout le temps et que les processeurs sont au maximum, installez plusieurs serveurs ou plusieurs processeurs (ou mise à jour processeur) (mais attention processeur inutile = dégradation)
  Note: IIS 4 sait tirer partie de deux processeurs mais pas au-dessus.
  (pour IIS 5 : 4 processeurs)

- Mettez a niveau votre cache L2: en ajoutant ou en mettant a niveau les processeurs en s'assurant d'être munis d'un cache secondaire (L2) important. Les applications de serveur de fichiers telles qu'IIS bénéficient d'un tel cache du fait que leurs chemins d'instruction impliquent de nombreux composants différents (cache de 1Mo a 2 Mo)

- Améliorez la gestion des DPC. Les plates-formes qui distribuent les interruptions sur tous les processeurs ne tirent aucun profit de l'affinité DPC par défaut de NT: mettez "ProcessorAffinityMax" à 0

- Ajoutez des adaptateurs réseau: si vous disposez d'un système multiprocesseurs qui ne procède pas à la distribution symétrique des interruptions alors ajoutez des cartes pour repartir le travail (association carte-processeur). Attention, comme tout matériel supplémentaire, de telles cartes provoquent une surcharge intrinsèque. Mais si un des processeur est 100% temps et plus de la moitié a faire du PDC, alors vous êtes sur la bonne voie.

- Limitez les connexions: si une mise à niveau des processeurs ou un ajout de processeurs n'est pas possible, réduisez le nombre maximum de connexions et procédez par étapes.

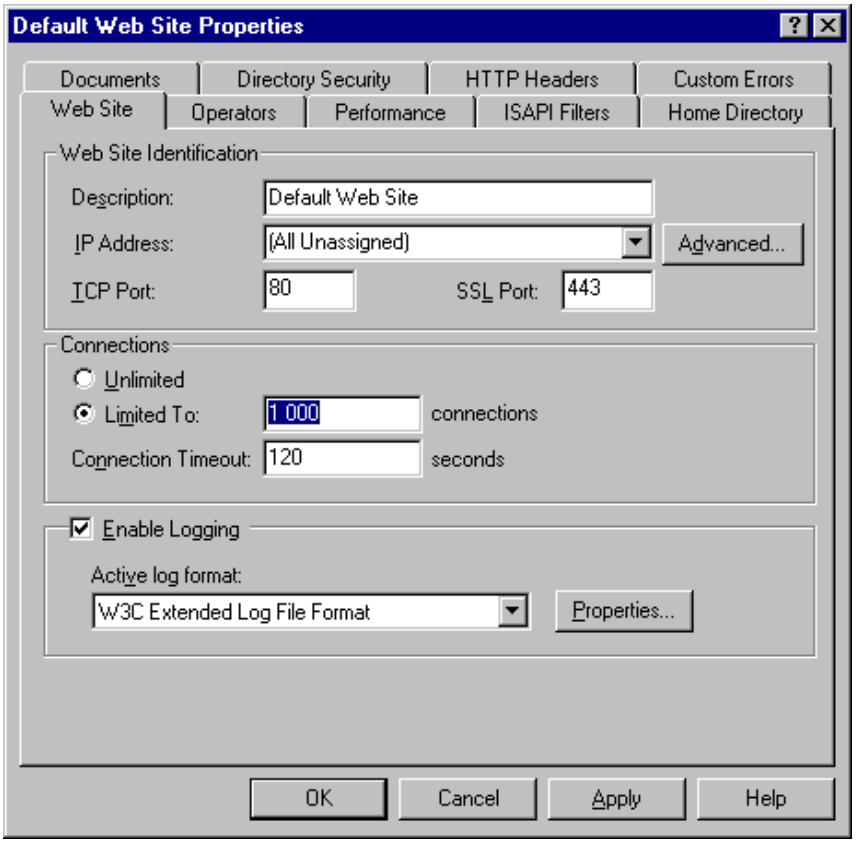


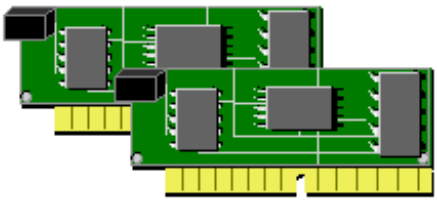
Figure 9 : Limitez le nombre de connexions si cela est envisageable

- Mettez à niveau les cartes réseau pour s'assurer de la prise en charge de la modération des interruptions.

- Réorganisez votre site Web:
  - Optimisez les scripts ASP
  - Optimisez les accès aux bases de données
  - Appelez des composants compilés depuis les pages ASP
  - Utilisez ISAPI au lieu d'ASP... et ASP au lieu de CGI
  - Favorisez les pages Web statiques et non dynamiques,
  - Diminuez l'utilisation de SSL
  - Remettez 'inproc' des applications fiables
  - Gérez au mieux la taille de vos images

Ajustez la longueur maximale de la file d'attente ASP et le nombre maximum de threads pour ASP (voir "ProcessorThreadMax" et "RequestQueueMax" dans la suite)

4. Le réseau



4.1. Les besoins

Bien évidemment, vu le travail d'IIS, le réseau est un point important…

Par réseau, on entend la carte ou les cartes mais également le câblage, les composants réseau, la charge à côté que doit supporter le réseau, etc..

Les deux facteurs à considérer sont: la bande passante effective du lien et la capacité de ce lien comme du serveur à prendre en charge les ressources du réseau.

Définitions:

- Bande passante = nombre d'octets échangés par le serveur, nombre de paquets, nombre de fichiers (c'est un concept assimilable à la vitesse du réseau)

- Capacité du réseau = capacité du serveur ET du lien de communication à acheminer le trafic ((c'est un concept assimilable au nombre de connexions simultanées que peut supporter le serveur)

Ces deux thèmes vont déterminer les deux principaux points à analyser dans ce chapitre.

4.2. Le réseau pour NT

a) La bande passante :

La bande utilisée par un serveur est le taux de transmission/réception des données.

Les compteurs perfmon sont à décliner suivant les couches OSI:
- compteurs sur les objets couche application: service WWW, service FTP, serveur SMTP…
- compteurs sur l'objet couche transport: TCP *
- compteurs sur l'objet couche réseau: IP *
- compteurs sur l'objet couche Liaison de données: Interface réseau *

Suivant la couche considérée, il pourra être observé ce qui est réellement mis sur le câble ou ce qui représente réellement le trafic du service (sans prendre en compte l'ajout des entêtes, le découpage en paquets, les retransmissions, les paquets de contrôle…)

Note: la présence de l'astérisque indique que ces compteurs ne s'activent que si le protocole SNMP est installé sur la machine

Les compteurs perfmon à employer sont:

- Les compteurs des Objets "Web Service", "FTP Service" et "SMTP Server" nommés "Bytes Sent/sec", "Bytes Received/sec" et "Bytes Total/sec"

  Rappel: ces compteurs mesurent les octets envoyés avant de scinder les paquets, les ajouts d'entêtes et de contrôle et ne tiennent pas compte des retransmissions. Ceci représente 60 à 70 % des données transmises !

- Objet "TCP : les compteurs "Segments" (segments envoyés, reçus, total, retransmis)
- Objet "IP": les compteurs "Datagrams" (datagrammes envoyés, reçus, total, retransmis)
- Objet "Network Interface": les compteurs "Octets" (octets envoyés, reçus, total).

Si la somme des octets transférés par les services Web est supérieure au 2 tiers de la bande passante du réseau, le goulot d'étranglement est le réseau.

b) La capacité du réseau

Si problème de bande passante intervient, les clients s'apercevront d'un problème avant le serveur qui, lui, continuera à établir des connexions, recevoir des demandes et transmettre des données.
Le client lui de son côté expérimentera des rejets, des dépassement délais d'attente, des réponses tardives…

Pour détecter cette situation il faut contrôler la réussite et l'échec des connexions établies par TCP: si la bande est suffisante, le serveur est à même d'établir et de satisfaire des connexions avant qu'elles ne dépassent le délais d'attente, sinon elles échouent.

Au niveau perfmon, les compteurs Service Web et service FTP n'afficheront que les connexions qui ont fonctionné, pas celles qui ont échoué... il faut donc se concentrer sur le compteur TCP.

Si un problème de bande passante est suspecté, cela s'accompagne en général d'un nombre de connexions TCP échouées conséquent. Le compteur "Connection Failures" de l'objet "TCP" s'apparente a ce symptôme. Il est à comparer avec le nombre de connexions établies et ré-initialisées.

Attention: ré-initialisée ne veut pas dire forcement problème car ce compteur inclus le nombre de navigateurs qui forcent une ré-initialisation pour minimiser la surcharge de la connexion en fermant de manière régulière des connexions par l'envoi d'un paquet TCP RESET(RST)

Si le compteur concernant les connexions établies arrive à une limite et ne bouge plus, vous avez atteint la limite de votre réseau. A partir de là, le nombre de connexions en échec va augmenter.

Attention: le compteur "Connexion établies" donne un chiffre à l'instant T alors que "Connexions échouées" et " Connexions ré-initialisées" montrent un total depuis le démarrage du service.

4.3. Le réseau pour IIS

a) Mesurer les performances

Le travail d'IIS consiste avant tout d'envoyer des fichiers. Chaque demande réussie correspond au minimum à l'envoi d'un fichier au minimum (bien souvent cela s'accompagne de plusieurs fichiers comme les images, les contrôles, les scripts, etc…)

Dans Perfmon, les objets Service Web et FTP ont des compteurs: nombre total de fichiers envoyés, reçus et transférés. Ce sont ces compteurs que vous devrez analyser pour déterminer les performances de votre serveur.

Attention: ces compteurs sont donc des totaux, pas des valeurs en cours.

Exemple ci-dessous: ce log a été généré alors que seulement une page HTM a été demandée. Mais cette demande s'accompagne d'une vingtaine de fichiers transférés.

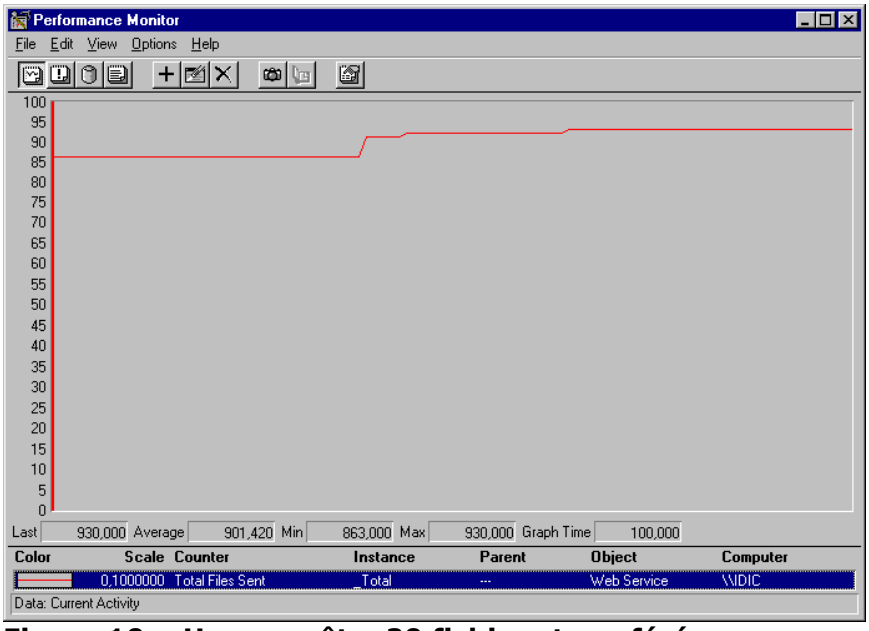


Figure 10 : Une requête, 20 fichiers transférés.

b) Le contrôle de la bande passante

Si la bande passante n'est pas suffisante par rapport à la charge demandée, vous pouvez envisager de mettre en place une limitation des pages HTML statiques (.html ou .htm). Cette limitation ne concerne pas asp, isapi ou cgi

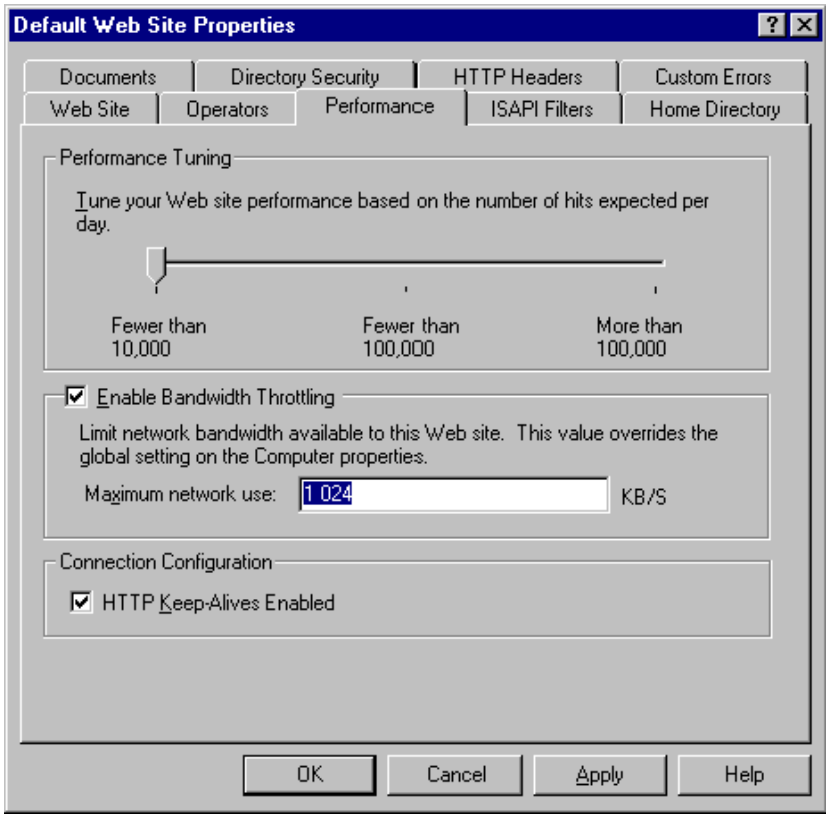


Figure 11 : Limitation de la bande passante.

Note: Vous pouvez également utiliser cette fonctionnalité pour simuler une liaison bas débit comme une liaison RNIS ou modem 33.6 par exemple.

Il n'est pas nécessaire de redémarrer suite à cette configuration, les effets sont directement visibles dans perfmon.

A l'activation de la limitation de la bande passante, IIS active un ensemble de compteurs dans perfmon. Pour les identifier, recherchez la présence des mots "E/S async." ("I/O Requests") dans les compteurs de l'objet "Internet Information Service global":

- Nombre de requêtes e/s asynchrones actuellement bloquées. ("Current Blocked Async I/O Requests")

- Nombre total de requêtes e/s autorisées, bloquées et rejetées. ("Total Allowed/Blocked/Rejected Async I/O Requests")

Et surtout, regardez le compteur "Measured Async I/O Bandwith Usage" qui est une mesure de l'utilisation de la bande passante par les e/s asynchrones (nombre d'octets/minute tel que l'indique un échantillonnage effectué par le limiteur de la bande passante):

- si ce compteur approche le nombre maximum défini dans l'interface, la limitation bloque les demandes de lecture (des pages statiques) mais autorise celles d'écriture et de transmission.

- si la largeur de bande employée dépasse la valeur maximale définie, la limitation rejette les demandes de lecture et bloque les demandes d'écritures et de transmission les plus importantes (seules les plus réduites sont autorisées).

4.4. Conclusion: Conseils d'optimisation concernant le réseau

- Accroissement de la bande passante effective des liens de communication

- Optimisation de la taille des fichiers graphiques (jpeg pour les photos, gif pour les graphiques)

- S'assurer du support des connexions http persistantes

- Ajustement de la longueur de la file d'attente de la connexion: par défaut quinze demandes, au delà c'est rejeté. Si le serveur rejette un grand nombre de connexions uniquement lorsque les services sont les plus actifs, vous pouvez accroître le nombre maximum d'éléments dans la file. C'est la valeur "*ListenBackLog*" sous

  HKEY_LOCAL_MACHINE \system\CurrencontrolSet\Services\InetInfo\Parameters

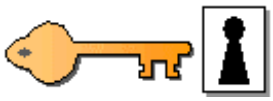
  Note: ne confondez pas cette valeur avec *ASPQueueTimeout* que nous verrons par la suite

- Détecter les routeurs en "trou noir" (pas de messages "ICMP Destination Unreachable" lorsque la taille du datagramme IP est trop longue pour le serveur destinataire et que le bit "ne pas fragmenter" est activé, entraînant la ré-initialisation de la connexion). Avec l'option "*EnablePMTUBHDetect*" à 1, le serveur va réagir à une transmission répétée sans accusé de réception en désactivant le bit de non-fragmentation. Une fois la transmission réussie, elle réduit la taille du segment et réactive le bit.

  HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\TCPIP\Parameters

  Attention: ce paramétrage concerne les trois services en même temps: WWW, FTP et SMTP

5. La Sécurité



5.1. Les besoins

Un serveur Internet ou intranet non sécurisé serait rapidement la proie de tous les hackers à la recherche soit d'informations confidentielles soit désireux de rendre le serveur indisponible.

Des moyens de sécurité efficaces entraînent une surcharge des performances parfois significative. Tout est encore une fois une question de juste équilibre entre le niveau de sécurité recherché et le sacrifice en performance que cela va entraîner

La seule manière de mesurer la surcharge sécuritaire consiste à comparer les performances avec la sécurité en place et sans… mais en considérant entre chaque test exactement la même configuration.

Il faut alors comparer:
- l'activité processeurs et la file d'attente des processeurs: l'authentification, la vérification des adresses IP, SSL: tout ça coûte beaucoup au processeurs
- la mémoire physique utilisée: le stockage et la récupération d'informations utilisateurs, les clés SSL: tout ça prend de la mémoire
- le trafic réseau. Par exemple l'augmentation du trafic entre le serveur et un contrôleur de domaine suite à la mise en place de NTLM (voir ci-après)
- la latence et les retards (un chiffrage met du temps: téléchargement de fichiers 10 à 100 fois plus lent)

5.2. La sécurité pour NT

Toute mise en place de sécurité entraîne des mécanismes d'identification, d'authentification et de vérification qui ont des conséquences évidentes en matière d'utilisation processeur et réseau.

Sur un serveur IIS, l'authentification peut se faire
- en anonyme
- en mode "authentification texte clair"
- en mode "stimulation/réponse" ou NTLM
- par des certificats
- via un annuaire (Site Server)
- et quantité d'autres méthodes standards ou non…

Chaque méthode a ses forces et ses faiblesses et, bien évidemment, plus une méthode est sûre, plus elle coûte en termes de performances.

Prenons l'exemple de NTLM. Ce mécanisme qui est le plus simple à mettre en place sur un environnement intranet et qui est bien plus sûr que le mode "authentification texte clair" entraîne un trafic réseau qui peut s'avérer important surtout si la base de données utilisateurs NT (SAM) sur lequel il s'appuie se trouve sur une machine distincte du serveur web. En effet, dans ce cas là, des communications vont devoir se

faire entre le serveur lui-même et un contrôleur de domaine pour autoriser ou refuser l'accès aux données du serveur.

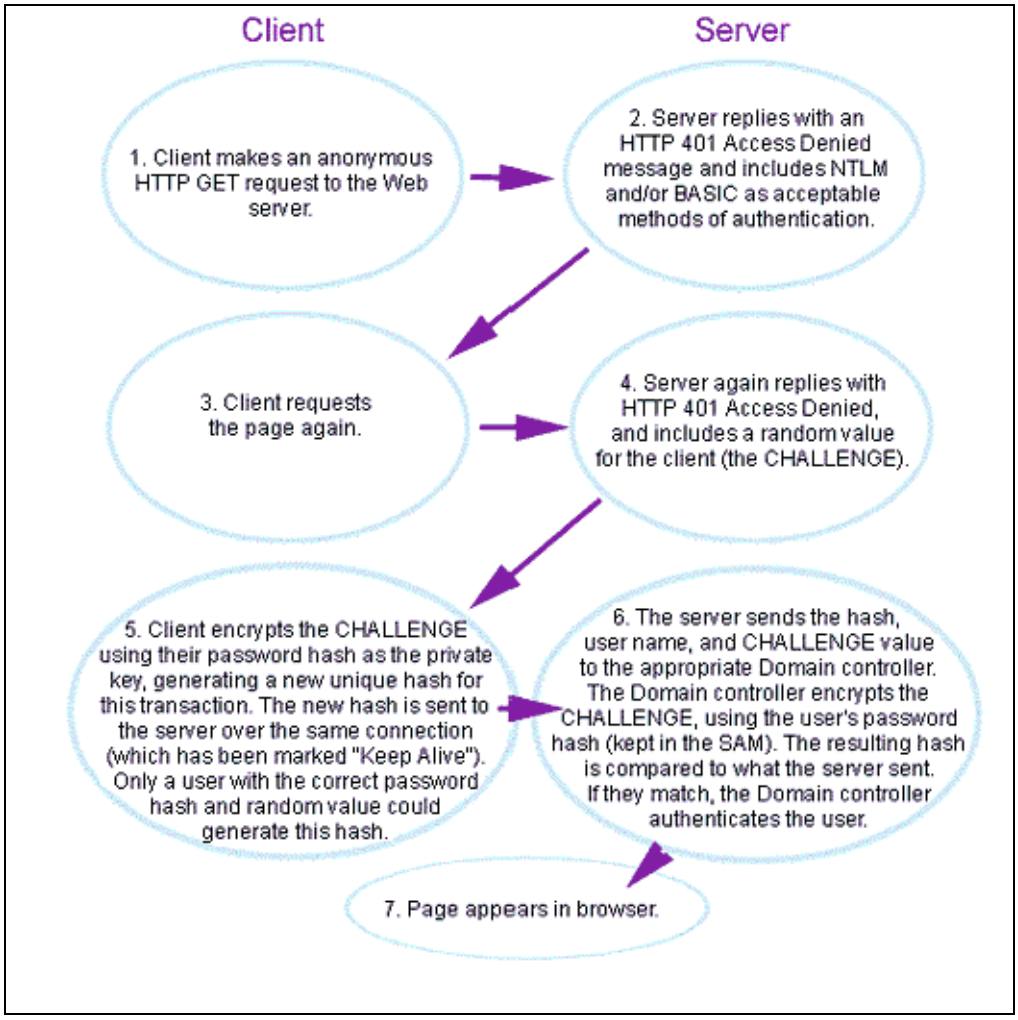


Figure 12 : Le mécanisme NTLM dans un contexte HTTP.

Le système de sécurité que vous choisirez et mettrez en place aura donc plus ou moins une influence sur les performances de votre site web.
Ce choix est d'autant plus important sur un serveur web où ce mécanisme sera joué autant de fois que de requêtes a sécuriser auront lieu, c'est à dire un grand nombre potentiel sur un serveur Internet. Assurez-vous de bien savoir ce qu'entraîne tel ou tel mécanisme sur tous les domaines (mémoire, processeur, réseau,…) avant de finaliser votre décision.

Si vous optez par exemple pour NTLM pour votre serveur intranet, sachez que le mécanisme Kerberos présente bien plus d'avantages en termes de performances pour un serveur IIS tout en garantissant un niveau de sécurité comparable ou supérieur.

5.3. La sécurité pour IIS

a) Stressez votre site

Les tests de stress sont les moyens les plus efficaces pour tester l'impact d'une configuration de sécurité sur un serveur Web.

Pour cela, il existe quantité d'outils de stress simulant des accès sur un serveur web. Microsoft a sorti dans ce domaine les outils suivants

- InetLoad
- InetMon (http://www.microsoft.com/siteserver/site/DeployAdmin/InetMonitor.htm)
- WCAT (http://download.microsoft.com/download/winntsrv40/Utility/1/NT4/EN-US/WCAT.EXE)
- Homer

Ce dernier ayant évolué dans le dernier outil de stress en date qui cumule par ses fonctionnalités les principaux atouts des outils cités ci-dessus: **WAS (Web Application Stress).**
Vous pourrez télécharger ce logiciel gratuit depuis l'adresse suivante: (http://webtool.rte.microsoft.com/) accompagnés de documents de présentation de l'outil et d'aide à l'utilisation.

Il vous faudra donc:
- établir une séquence de requêtes standard qui seront représentative de l'activité de votre serveur (le script)
- configurez votre serveur dans une première configuration
- jouer le script
- examiner les performances de votre serveur par perfmon
- puis changez la configuration du serveur
- rejouer le script
- examiner les performances à nouveau et examinez les différence observées.

Puis changez de script pour évaluez les conséquences de votre choix suivant la charge demandée…

Note: Bien sûr, cette procédure est adaptable pour tous les paramètres déjà vus au cours des précédents chapitres et ceux à venir.

b) Quels compteurs examiner ?

Il faut surveiller les connexions anonymes et non anonymes pour déterminer le nombre d'utilisateurs qui ont accéder aux ressources sans s'identifier et ceux qui ont eu besoin de s'identifier pour accéder aux données.

Il convient de restreindre au maximum les ressources à sécuriser pour entraîner un minimum de surcharge liée à la sécurité.

Avec perfmon vous pourrez déterminer si, sachant les données fournies part votre serveur, vous sécurisez juste ce qu'il faut ou si vous protégez des données qui n'ont pas besoin de l'être.

Les compteurs à regarder sont associés aux objets du service Web (on retrouve les même pour le service FTP)
- "nombre d'utilisateurs anonymes/sec" et "nombre d'utilisateurs non anonymes/sec"
- "nombre actuel/maximal/total d'utilisateurs anonymes" et "nombre actuel/maximal/total d'utilisateurs non anonymes".
- "Nombre d'utilisateur anonymes en cours" et "Nombre d'utilisateurs non anonymes en cours". **Attention:** ceux-ci affichent les connexions et non les utilisateurs. Les utilisateurs qui se connectent plus d'une fois sont comptés à chaque fois qu'ils se connectent. D'autre part, les tentatives de connexions qui ont échouées ne sont pas comptées.

Les données ainsi recueillies doivent alors être combinées avec les compteurs processeurs, la file d'attente des processeurs, mémoire, lecture et écriture sur disque, etc.… pour évaluer les performances.

Remarque: Vous pouvez également utiliser perfmon pour le décompte des erreurs "non trouvé". En effet, ce compteur va certes décrire les demandes de clients non satisfaites parce qu'elles faisaient référence à une page web ou à un fichier qui n'existe pas mais également va inclure les tentatives d'accéder à des documents qui lui sont interdits.

c) Au niveau des composants compilés

La sécurité est également un point à ne pas mettre de côté lors de la création de composants personnels tels que des contrôle ActiveX ou des filtres ISAPI.

Il n'est pas rare de voir des problèmes liés à l'impersonation qui entraînent des accès refusés en quantité avant d'autoriser ou refuser définitivement un accès et ainsi avoir des conséquences néfastes sur les performances de votre serveur.

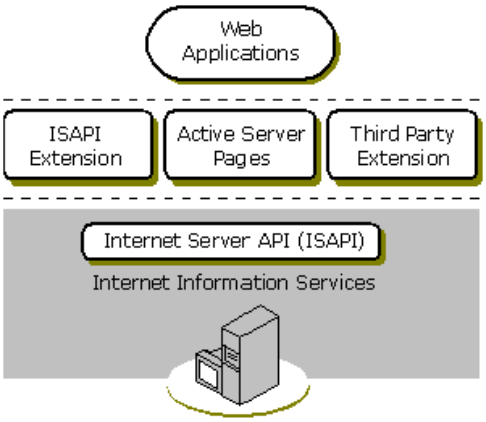
Il convient de bien maîtriser les mécanismes de sécurité liés à COM par exemple et les questions d'impersonation, de délégation et d'authentification pour écrire au mieux son composant et en assurer ses performances.

5.4. Conclusion: Conseils d'optimisation concernant la sécurité

Configurez le système pour gérer la sécurité puis si cet ajout de sécurité demande plus de ressources vous pouvez:
- Mettre de la sécurité seulement où cela est nécessaire
- Mettre à niveau ou ajouter des processeurs (SSL est gourmand)
- Mettre à niveau le cache des processeurs (cache L2 important jusqu'à 2 Mo)

- Ajouter de la mémoire. **Par contre, augmenter la capacité disque ne sert pas a grand chose: la sécurité est surtout une affaire de processeurs et de mémoire.**
- Limiter les accès réseau (NTLM entraîne des appels aux DCs)

6. Les Applications



Si les précédents chapitres vous permettront de paramètrer au mieux votre système vis à vis de la charge, rien ne remplacera une optimisation de vos applications. Les performances gagnées en modifiant une application et en améliorant ses performances n'ont en général aucune commune mesure avec celles gagnées par l'ajustement du système (excepté lors d'un réel goulot d'étranglement).

Imaginez une opération (telle qu'une requête sur une base de données) qui demande une seconde pour s'exécuter. Dans un environnement intranet, avec une centaine d'utilisateurs, cette durée ne porte pas à conséquences. Mais avec un serveur Internet, où le nombre d'utilisateurs potentiels est bien plus important, cela peut conduire à des latences non acceptables. Nous verrons dans ce chapitre comment ajuster la taille de la file d'attente des requêtes ASP pour ainsi refuser des nouvelles demandes quand le nombre de requêtes en attente multiplié par le temps du cycle d'exécution moyen d'une requête dépasse un seuil fixé à l'avance. Ce seuil dépendra de la capacité du système à gérer plus ou moins rapidement un nombre donné de requêtes simultanées.

Nous verrons donc dans ce chapitre les principaux ajustements possibles mais, bien évidemment, réduire dans notre exemple le temps de la requête à 10 milli-secondes conduira à un gain bien plus important que la modification de tous les paramètres NT ou IIS existants.
Ce chapitre ne s'appliquera pas a vous donner des conseils en développement pur: non seulement cela conduirait a des considérations très diverses (base de données, modèle de composants, langage utilisé, …) mais demanderais des pré-requis qui sortent du cadre de ce document. Nous nous attacherons à des considérations classiques et examinerons en priorité l'aspect système de l'optimisation d'un serveur IIS.

6.1. Les compteurs Perfmon

Quels sont les compteurs de l'analyseur de performances à regarder pour analyser le comportement d'une application ?

Compteurs décomptant les requêtes ASP: ils affichent tous la dernière valeur observée et non une moyenne à l'exception du compteur "nombre de requêtes par seconde"
Ils se retrouvent tous dans l'objet "Active Server Pages", vous y trouverez des compteurs relatifs au nombre de requêtes en cours d'exécution, au nombre de requêtes mises en file d'attente, au nombre de requêtes exécutées, au nombre de requêtes par seconde, au nombre de requêtes échouées/non autorisées/non trouvées/rejetées/hors délais et au nombre total de requêtes.

Compteurs applications ISAPI et CGI: affichent également tous la dernière valeur observée et non une moyenne.
Ils sont présent dans l'objet "Service Web" et traitent le ombre de requêtes extensions ISAPI/CGI en cours, le nombre maximal de requêtes d'extensions ISAPI/CGI et le nombre total de requêtes d'extensions ISAPI/CGI.

Le travail consiste à examiner ces compteurs et observer leur évolution suivant la charge demandée.

6.2. WAM

L'architecture applicative d'IIS 4.0 est basée sur le Web Application Manager (WAM). C'est un wrapper COM pour toute fonction ISAPI existante. Il est implémenté sous la forme d'un objet COM enregistré via MTS, c'est à dire qu'il encapsule tous les appels pour localiser, charger et exécuter une requête ISAPI.
De par la conception d'IIS, chaque application IIS a son objet WAM associé responsable de son fonctionnement ISAPI.

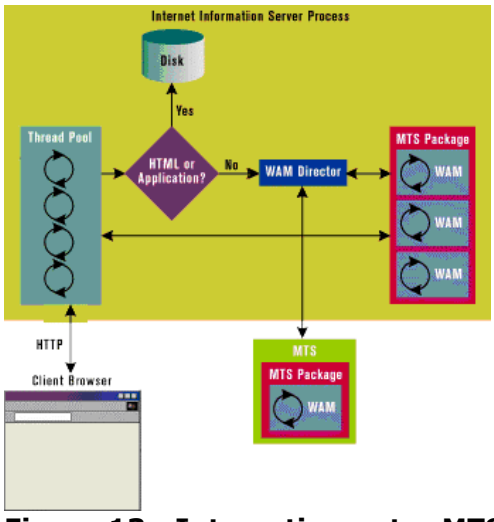


Figure 13 : Interaction entre MTS, WAM ET IIS.

Par défaut les applications ASP et ISAPI s'exécutent dans le processus IIS: Inetinfo.exe.
Puisque WAM est un composant de MTS, il peut tirer avantage de l'isolation des processus: vous pouvez faire exécuter vos applications dans leur propre espace mémoire séparé (au sein d'un mtx.exe) pour ainsi les observer plus spécifiquement.

Note: De plus, en isolant vos application, vous tirez également avantage de FailFast, mécanisme d'arrêt et de relance des processus mtx.exe s'apparentant à de la tolérance de panne.

En conséquence, prenez garde à bien observer inetinfo.exe ET tous les éventuels mtx.exe présents sur votre système. Au niveau de perfmon, cela vous obligera a ajouter toutes les éventuelles instances de mtx.exe lorsque vous observerez des compteurs associés à l'objet "Processus"

La figure suivante décrit cette remarque, la prise d'un log perfmon a permis de déterminer qu'une instance de mtx.exe drainait l'activité processeur d'un serveur.

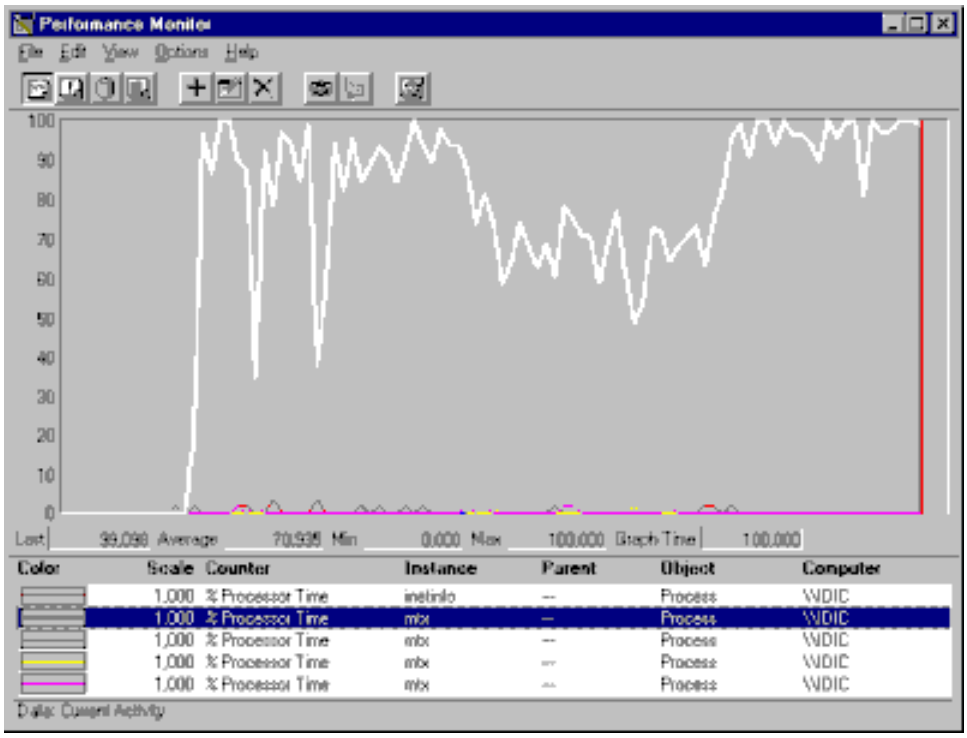


Figure 14 : L'analyseur de performance et les mtx.exe

Attention: Mettre une application dans un espace mémoire séparé (dans un mtx.exe) entraîne un coût en performance. Ce coût est généralement peu important, mais il est bon de savoir qu'il existe.

Note: Pour plus d'informations sur mtx.exe et surtout pour savoir associer l'instance d'un mtx.exe avec une application web, merci de vous procurer un autre livre blanc, du même auteur, "Internet Information Serveur 4.0: Procédures de résolution d'incidents bloquants" (Janvier 2000).

6.3. Les principaux ajustements

a) Ajustement de la réserve de threads et de la file d'attente ASP

En cas de charge importante, les scripts ASP peuvent créer une file d'attente ASP (Exemple: un script appelle un composant qui reçoit plus d'appels qu'il ne peut en gérer).
Dans un tel cas, les demandes reçues par la page ASP appelant le composant sont placées dans une file d'attente FIFO d'une longueur de 500 requêtes par défaut.

Si les conditions de blocage ne durent que quelques secondes, la mise en file d'attente va aplanir les variations de la charge et ainsi toutes les demandes reçues seront satisfaites de manière convenable (particulièrement s'il existe suffisamment de threads pour contrôler les autres demandes (non ASP) au fur et à mesure de leur arrivée).

Si par contre le blocage dure pendant mettons 30 secondes, un utilisateur dont la demande est vers la fin de la queue peut attendre longtemps (par exemple 15 secondes) avant de voir sa demande satisfaite. Ce temps d'attente peut l'entraîner à cliquer sur le bouton de rafraîchissement de la page demandée et ainsi à la création d'une autre requête qui va être mise en attente, etc…
Le syndrome du bouton "Refresh" peut donc aggraver la situation.

Il convient donc de réduire au mieux ce temps de blocage puis d'ajuster la longueur de la file d'attente.

- Pour le temps de blocage, la paramètre est **ProcessorThreadMax:**

Ce temps de blocage est représentatif de la capacité de votre serveur a répondre plus ou moins vite à une charge importante. Il est donc directement lié au nombre de threads et à la capacité processeur du serveur.

Il n'est pas possible de calculer quel devrait être le nombre exact de threads adéquats mais les considération suivantes vous permettrons de vous donner une idée de la quantité de threads que votre serveur devrait créer en fonction de sa charge et de ses capacités.

Si le compteur de l'objet ASP: "nombre de requêtes actuelles mises en file d'attente" n'est pas important (pratiquement 0) ou décroît toujours, et que l'utilisation processeur est faible, alors tout est bien dimensionné, ne touchez à rien.

Si la file croit et décroît et que vos processeurs s'exécutent au dessous de 50% alors certaines requêtes provoquent des blocages et vous profiterez sans doute d'une augmentation du nombre de threads. Commencez par modifier la valeur *ProcessorThreadMax* dans la base de registre de 10 à 20 puis redémarrez l'ordinateur. Attendez-vous alors à une augmentation de l'utilisation des processeurs mais la longueur de la file devrait en contre partie se modifier plus rapidement lors de pics d'activités.

La valeur *ProcessorThreadMax* se trouve sous la clé de la base de registre suivante: *HKEY_LOCAL_MACHINE\System\currentControlSet\Services\W3SVC\ASP\Parameters*

Si après cet ajustement la file continue a s'allonger et l'utilisation des processeurs décline, il se peut que votre serveur éprouve de sérieux problèmes de blocage (exemple: les threads attendent la réponse d'une ressource externe telle qu'une base de données). De même si il y a un bogue dans un composant, il aura tendance a se manifester plus rapidement vu qu'il y a plus de threads.

Si par contre, la longueur de la file reste insignifiante et l'utilisation des processeurs s'accroît, vous pouvez continuer a augmenter *ProcessorThreadMax*.

Conseil: Ne dépassez jamais les 70% d'utilisation CPU.

- Pour la longueur de la file d'attente ASP, le paramètre est **RequestQueueMax:**

Vous pouvez calculer le temps de réponse en vérifiant combien de temps est nécessaire à la file d'attente pour revenir à zéro après qu'elle ait atteint sa valeur maximale (sur un site fluide et sans blocage, elle demeurera la plupart du temps proche de zéro)

Votre manipulation consistera en l'ajustement de la valeur *RequestQueueMax* sous *HKEY_LOCAL_MACHINE\\System\currentControlSet\Services\W3SVC\ASP\Parameters*

Tout d'abord, déterminez vote temps de réponse cible (par exemple 10 secondes) et conservez à la file d'attente une taille telle qu'elle ne renfermera jamais plus de quelques secondes de travail. La taille idéale va correspondre à être en dessous du temps de réponse maximum admissible mais au dessus de la taille type d'attente de la file au moment d'une pointe d'activité.
Une trop faible taille entraînera trop d'erreurs "Serveur occupé"
Une trop grande valeur fera que vous rencontrerez plus d'utilisateurs perdant patience lors du traitement d'un requête et qui vont abandonner (donc du travail serveur pour rien) ou faire un "Refresh" et rendre la situation encore pire.

Si vous ne disposez pas de données, optez pour un réglage initial au taux de 1/1 entre la longueur de la file et le nombre total des threads: par exemple avec un *ProcessorThreadMax* de 25 avec quatre processeurs, commencez par un *RequestQueueMax* de 100 puis procédez ensuite à vos ajustements.

b) Augmentez le nombre de moteurs de scripts mis en cache

Cette manipulation est à effectuer de préférence sur les "Master Properties" qui sont l'endroit ou sont définies les propriétés par défaut dont vont hériter tout nouveau site web créé ou tout site existant ne spécifiant pas sa propre valeur pour un paramètre. (concept d'héritage de la metabase)

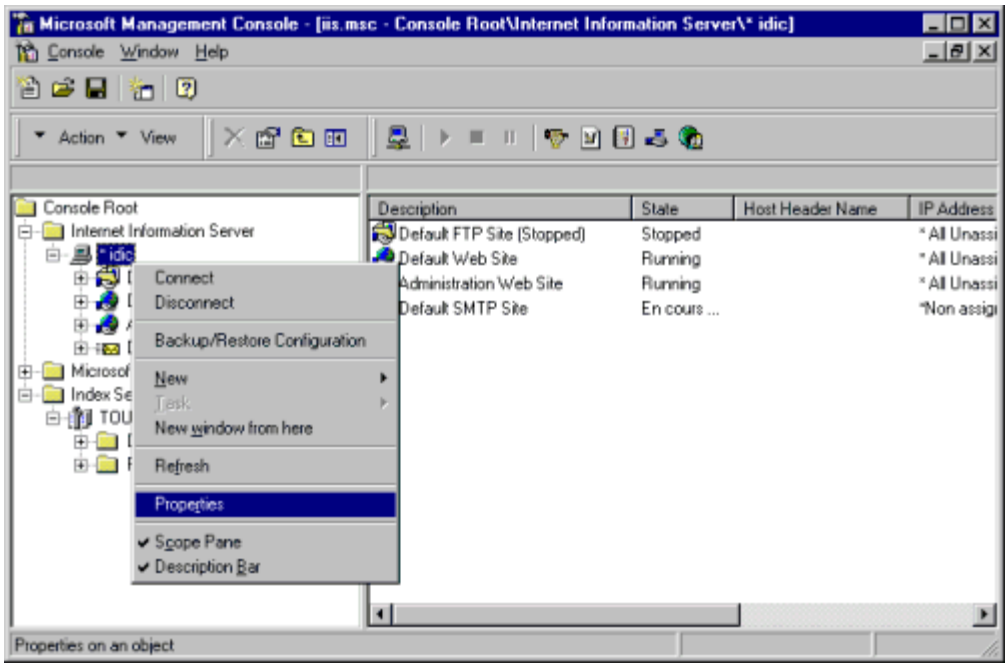


Figure 15 : Afficher les propriétés principales (1)

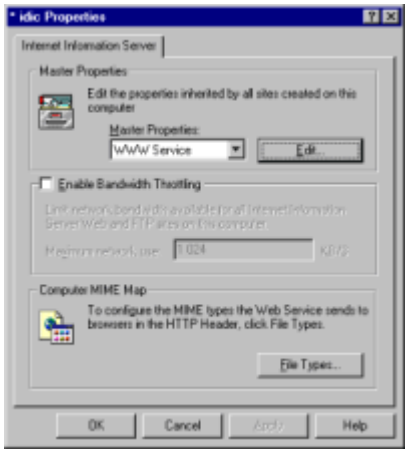


Figure 16 : Afficher les propriétés principales (2)

Elle consiste à augmenter le nombre de moteurs de scripts mis en cache. En effet, il y a compilation de script et création d'un moteur de script à chaque requête faisant appel à VBScript, JScript, etc… Modifier cette valeur permet d'ajuster le nombre de moteurs mis en cache par rapport au nombre attendu de requêtes faisant appel à des scripts.

Pour cela, affichez l'onglet "Options du processus" et entrez la valeur désirée.

Remarque: cet onglet n'apparaîtra que si l'objet dont vous affichez les propriétés par la MMC est donc un processus. C'est le cas pour les propriétés principales (processus inetinfo) ou une application web tournant dans un espace mémoire séparé.

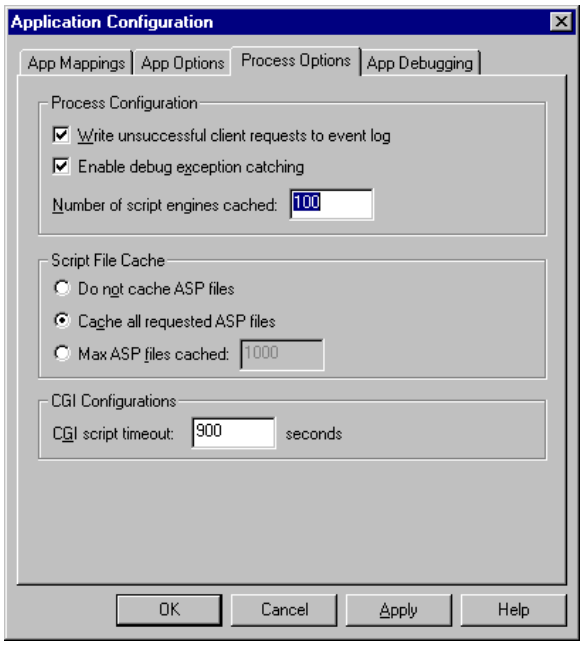


Figure 17 : Fixer le nombre de moteurs de scripts mis en cache

c) Ajustez le nombre de fichiers ASP mis en cache

Cet ajustement est également à faire de préférence sur les propriétés principales. IL se fait sur le même onglet que pour le paramètre précédent (donc même remarque).

Par défaut, toute page ASP est cachée, modifier ce paramètre permet soit d'économiser de la mémoire, soit éviter de faire demander un rafraîchissement de pages asp dont le résultat est constamment mis à jour (cours de la bourse par exemple).

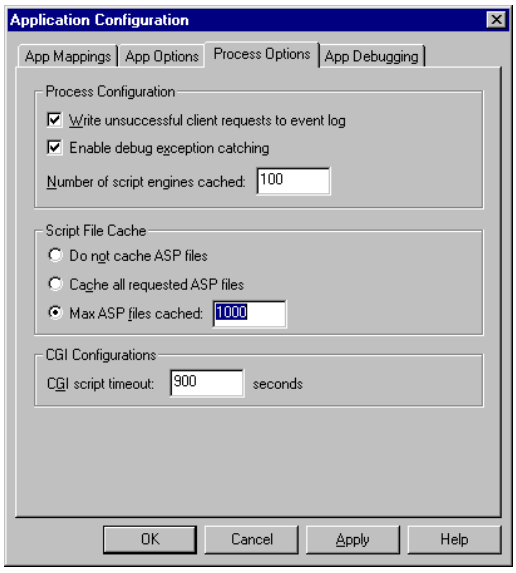


Figure 18 : Fixer le nombre de pages ASP mises en cache

d) Ajustez le 'timeout' des scripts ASP

Toujours à effectuer à effectuer de préférence sur les propriétés principales, la modification de ce paramètre permet de modifier la durée à partir de laquelle une requête ASP est considérée comme demandant trop de temps pour son exécution. Si ce temps est dépassé, la requêtes est annulée et le client s'en trouve notifié.

Cette valeur dépend donc de la réactivité du code implementé dans vos pages asp et des ressources auxquelles ce dernier fait appel (base de données extérieure, fichiers sur un autre serveur, etc…)

Par défaut, un script ASP dispose de 90 secondes avant d'être annulé par le serveur. Passer cette valeur à 30 secondes est courant. Cela permet de décharger le service web des requêtes qui de toutes les façons n'aboutissent visiblement pas.

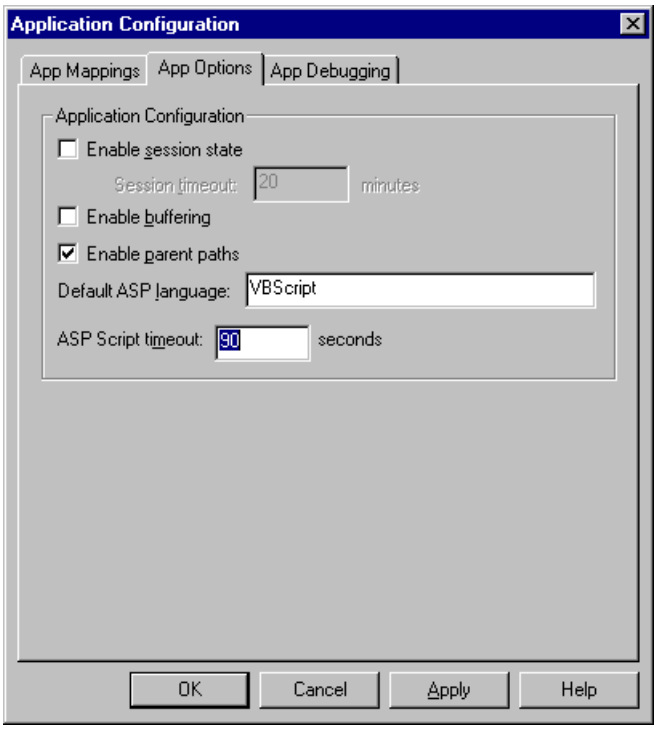


Figure 19 : Fixer le délais d'exécution autorisé pour les pages ASP

Note: l'ajustement de ce paramètre à priori anodin conduit dans certains cas à des gains de performances non négligeables.

e) Paramétrage et metabase

Les ajustements suivants correspondent à des valeurs dans la metabase. Certains ont leur équivalent dans la MMC mais d'autres ne pourront être modifié qu'en modifiant la metabase elle-même directement.

Vous ne pourrez procéder donc à ces ajustements que via MetaEdit (équivalent de RegEdit pour la base de registre), des scripts ADSI ou AdsUtil.vbs.
Ce dernier utilitaire est fournit en standard avec l'option pack à l'installation et se trouve dans le répertoire %SYSTEMROOT%\system32\inetsrv\adminsamples.

Par exemple, pour changez le délais d'exécution des requêtes ASP comme nous en avons discuté dans le paragraphe précédent, changez la valeur de "ASPQueueTimeout" à 30 sec (0 indique un temps illimité, par défaut il est de 90 secondes)
Pour cela entrez la commande suivante à partir d'une invite de commande:

```
adsutil set w3svc/1/AspQueueTimeout 30
```

Deux autres paramètres peuvent être modifiés de la même façon il sagit de:

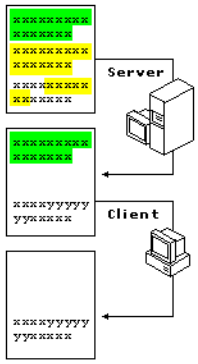
- ServerListenBackLog: dicte le nombre de connexions en attente maximum. Par défaut cette valeur est à 200, une valeur de 0 indique un nombre illimité, une valeur courante est 500 pour un serveur internet.

  ```
  adsutil.vbs set w3svc/1/ServerListenBackLog 500
  ```

- MaxEndPointConnections: qui définit le nombre de sockets par port. Par défaut cette valeur est à 200, une valeur de 0 indique un nombre illimité, une valeur courante est 500 pour un serveur internet.

  ```
  adsutil set w3svc/1/MaxEndPointConnections 500
  ```

6.4. Les scripts côtés client



Selon les fonctionnalités désirées, il peut être judicieux de déplacer du code serveur vers les clients. Il arrive qu'un calcul soit fait au niveau du serveur alors que le client pourrait très bien le faire ! Pourquoi alors charger inutilement le serveur ?

Donc, sachez faire la part des choses et déportez le plus de temps CPU (et d'utilisation mémoire) possible du côté du client.

Pour des menus dynamiques par exemple, sachez tirer partie des événements HTML 4.0 au lieu de procéder à des recalculs de pages côté serveur.

Attention: Ne confondez pas "événements HTML 4.0" et DHTML; ce dernier est une norme Microsoft (donc fonctionne avec Internet Explorer). De plus, il faut avoir conscience que ces événements ne fonctionnent qu'avec des navigateurs récents.

6.5. Les moteurs de script

Le choix entre VBScript et JScript tend à prendre une importance de moins en moins grande pour vos scripts. Il est tout de même conseillé de préférer VBScript pour le code s'exécutant sur le serveur (ASP) et JScript pour le code côté clients.

Quelque soit votre choix, il est fortement conseillé de mettre à les moteurs de scripts utilisés avec la dernière version disponible (Microsoft Windows Script 5.1 au moment de la rédaction du document présent).

Pour télécharger les moteurs de script à jour: http://msdn.microsoft.com/scripting/

6.6. Gérer au mieux son cache

Mettre en cache des données qui ne varient pas souvent est un facteur d'optimisation non négligeable.

Il y a deux types de mise en cache à pratiquer lors de l'écriture d'une application:

Le "Output Caching': qui consiste à mettre des objets HTML déjà formatés (avec les Tags HTML et les données) dans des variables applications. Ainsi, le premier utilisateur d'une application entraînera le calcul d'éléments HTML dont le résultat sera utilisé tel quel pour tous les autres utilisateurs de l'application.

Le "InPut Caching": qui consiste à mettre en cache un ensemble de données, même si certaines ne seront pas utiles pour la requête spécifiquement… elles le seront plus tard probablement. Par exemple, vous pouvez envisager de stocker dans une variable application un recordset complet bien que toutes les données de ce recordset ne soit pas utiles pour satisfaire la requête actuelle. Les autres utilisateurs de l'application web bénéficieront de la variable sans avoir le coût de la récupération des données dans la base.

La figure suivante montre un code ASP qui combine ces deux concepts en récupérant un ensemble de données se situant sur une base de données pour construire une combo box et stocker ensuite le code HTML de ce composant dans un variable application. A chaque exécution de la page, la variable application est analysée et si elle est non vide, le calcul n'est pas refait.

```
<%@ LANGUAGE=JavaScript %>

<html>
<body>

<form method=post>
Quel est votre nom ? <%= getUsersListBox() %>
<p>
<input type=submit>
</form>

</body>
</html>
```

```
<%

function getUsersListBox()
{

  UsersListBox = Application("UsersListBox")

  if (UsersListBox != null) return UsersListBox

  crlf = String.fromCharCode(13, 10)
  UsersListBox = "<select name=Users>" + crlf;
  SQL = "SELECT EmailAddress FROM Users ORDER BY EmailAddress";
  cnnUsers = Server.CreateObject("ADODB.Connection");
  cnnUsers.Open("Users", "WebUser", "WebPassword");
  rstUsers = cnnUsers.Execute(SQL);
  fldEmailAddress = rstUsers("EmailAddress");
  while (!rstUsers.EOF){
    UsersListBox = UsersListBox + "  <option>" +
      fldEmailAddress + "</option>" + crlf;
    rstUsers.MoveNext();
  }
  UsersListBox = UsersListBox + "</select>"
  Application("UsersListBox") = UsersListBox
  return UsersListBox;

}
%>
```

Figure 20 : Cacher des données autant que possible

GLOBAL.ASA

```
<!--METADATA TYPE="TypeLib"
   FILE="C:\Program Files\Common Files\system\ado\msado15.dll"-->
<SCRIPT LANGUAGE=VBScript RUNAT="Server">
Sub Application_OnStart
  SQL = "SELECT CompanyName, City FROM Customers"
  cnnAdvWorks = "DSN=AdvWorks"
  Set rsCustomers = Server.CreateObject("ADODB.Recordset")
  ' This is usable disconnected
  rsCustomers.CursorLocation = adUseClient
  rsCustomers.Open SQL, cnnAdvWorks, adOpenStatic, AdLockReadOnly
  ' Disconnect the Recordset
  rsCustomers.ActiveConnection = Nothing
  Set Application("rsCustomers") = rsCustomers
End Sub
</SCRIPT>
```

Customers.ASP

```
<%
  Set myCustomers = Application("rsCustomers").Clone
  Set CompanyName = myCustomers("CompanyName")
  Set City        = myCustomers("City")
  Do Until myCustomers.EOF
%>
<b><%= CompanyName %></b> is located in <b><%= City %></b>.<p>
<%
  myCustomers.MoveNext
  Loop
%>
```

Figure 21 : Un autre exemple avec une simple application de deux pages (page principale et Global.asa)

6.7. Optimisez votre code ASP

Bien que le sujet de ce livre blanc ne soit pas de donner des conseils de pur codage de vos applications web, les points suivants constituent les bases à garder à l'esprit lors de l'écriture de pages ASP:

- Utilisez la bufferisation : <% Response.Buffer = True %>
  Ceci permet d'optimiser le flux des données envoyé au client et d'utiliser Response.Redirect au milieu de votre code.

- Utilisez un Response.Write à la place de plusieurs <%= ... %>

- Supprimez les remarques HTML (toujours pour optimiser le flux)

- Faites des Set <Objet> = nothing dés que possible pour libérer la mémoire.

- Evitez l'utilisation des variables 'Server'. Accéder aux 'server variables' oblige le serveur Web a procéder à une requête spéciale et à collecter toutes les variables serveur.

- Utilisez les dictionnaires via Scripting.Dictionary (non compatible avec les objets "Session" ou "Application" car 'Apartment-Threaded')

- Pour des objets dont l'utilisation n'est pas obligatoire, utilisez le tag <Object> plutôt que Server.CreateObject (L'objet est alors créé immédiatement)

- Utilisez des fichiers GLOBAL.ASA et stockez dans des variables applications des objets propres à l'application (comme les fichiers par exemple) (voir également la figure 21)

- Utilisez des variables locales en priorité par rapport à des variables globales (les première sont résolues à la compilation, les secondes à l'exécution)

- Ne-redimensionnez pas des tableaux, quitte à sur-dimensionner une fois pour toute (préférence de la vitesse par rapport à la mémoire)

6.8. Les composants COM

Les deux principaux avantages liés à l'utilisation de composants compilés sont:
- Ré-utilisabilité
- Performance

De plus, vous séparez alors le plus possible le code de la présentation et ainsi vous facilitez le 'debugging' de vos pages.

Les règles de base à appliquer dans le développement de composants COM sont:
- Appuyez-vous sur le cache processeur, sur les machines multiprocesseurs.
- Mettez en cache judicieusement: les méthodes de mises en cache sont multiples. Le constat est toujours le même: sacrifier de la mémoire pour gagner du temps.

- Créez un minimum de threads (diminuez les 'context switches')
- Allouez/Libérer un minimum d'objets: allouer des blocs mémoire coûte cher, les libérer peut coûter encore plus cher.
- Minimisez les dépendances externes: tout ce qui n'est pas sur le même ordinateur que votre site représente un goulet d'étranglement potentiel.
- Vérifiez que chaque composant peut s'exécuter plus rapidement que le plus court délais auquel il est appelé: si un composant doit être appelé vingt fois par seconde, il doit compléter chaque cycle en bien moins de 1/20 de seconde ou il se bloquera. Le blocage d'un seul composant peut anéantir les performances de votre site en totalité.

6.9. Les variables sessions et applications

ASP dispose de six objets de base: Request, Response, BrowserType, Server, Session et Application. Ces deux derniers permettent de maintenir des informations entre différents appels de pages.

L'objet Session vous permet de stocker des informations concernant un utilisateur.

L'objet Application vous permet de stocker des informations concernant une application et qui sont donc partagées entre les utilisateurs de cette application.

IL FAUT UTILISER CES DEUX OBJETS AVEC DISCERNEMENT. IL NE FAUT PAS STOCKER D'OBJETS DANS DES VARIABLES DE SESSION.

Stocker un objet dans une variable session entraîne un coût mémoire d'autant plus important qu'il y a d'utilisateurs. Sur un site Internet où le nombre d'utilisateurs est potentiellement infini, ce coût peut s'avérer énorme et conduire à un épuisement de la mémoire disponible.

En plus des considérations de mémoire, il y a d'autres considérations relatives aux types d'appartements des threads (MTA/STA) et aux modèles de threads des composants COM (Both, Free, Single et Apartment):

- Lorsqu'un composant COM est activé par un thread, COM regarde le modèle du composant (stocké dans le registre). Suivant ce type et suivant si il est compatible ou pas avec le type d'appartement du thread, le composant va tourner dans le thread appelant ou un autre thread va être créé pour le faire tourner.
- Le moteur ASP d'IIS créé uniquement des threads en type STA
- Tous les composants COM créés en VB étant basés sur le modèle "Apartment", ils vont donc être lancés dans le même thread qui va l'appeler. Si c'est ASP qui fait appel à ce type de composant, le composant va donc tourner dans le thread d'ASP.

Au sujet des variables de session d'ASP et les composants STA (modèle de thread "Apartment") (donc le cas des composants de VB):
A chaque fois qu'un objet est stocké dans une variable de session, ASP va se souvenir du numéro du thread qui a mis l'objet dans cette variable. C'est ce thread qui sera toujours utilisé pour alors accéder à l'objet session ainsi créé. En conséquence, des composants VB utilisés dans des pages ASP peuvent présenter des fonctionnements "bizarres" au premier abord concernant surtout la sécurité d'accès à des ressources.

Pour la même raison, des différences de comportements peuvent se produire. Comportements visibles surtout lorsqu'on passe une application web de 'inproc' en 'out of process'.

Les variables applications d'ASP et les objets COM:
Stocker un objet COM dans une variable application entraînerait des problèmes de disponibilité du thread associé à cette variable (c'est effectivement un comportement similaire au précédent paragraphe qui a lieu). Si cet objet bloque, tous les utilisateurs de l'application seront en attente de la libération du thread.

Si on utilise pas les objets session, alors comment s'en passer ?
- Utiliser des bases de données (ADO) pour stocker des informations relatives à une session
- Utiliser des cookies
- Utiliser des certificats (uniquement pour l'identification)
- Passer les données avec l'URL (Par Get ou Post)
- …

Le fait de choisir de se passer des variables Sessions vous permettra de désactiver la gestion des Sessions:
- Par l'interface
- Par code ASP: <% @ENABLESESSIONSTATE=FALSE %>

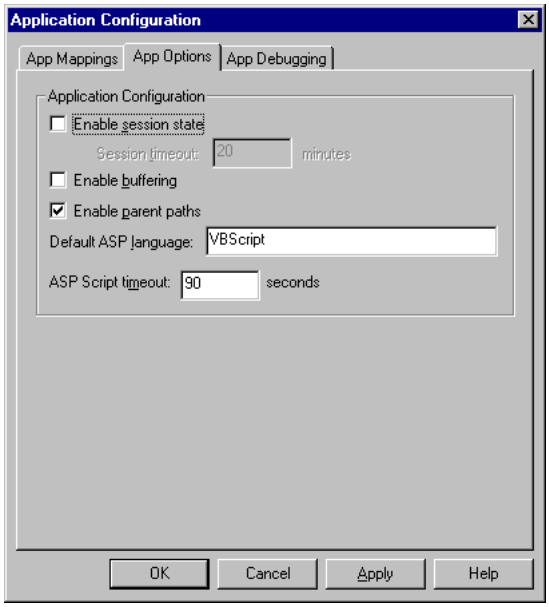


Figure 22 : Désactiver la gestion des sessions

6.10. Les accès aux bases de données

Les accès aux bases de données sont les accès les plus courants effectués dans une page ASP et sont également les raisons les plus courantes des limitations de performances.

Il faut savoir que l'optimisation des requêtes est un très important moyen d'optimisation des pages de vos serveurs. Vous pouvez très certainement gagner en performance uniquement en optimisant vos requêtes SQL.

L'utilisation du "Connection pooling", le fait de ne pas passer par des bases temporaires, de bien typer ses curseurs, de n'utiliser des curseurs que lorsque c'est vraiment nécessaire, de passer par OLEDB et non plus via ODBC et quantité d'autres considérations vous permettrons de gagner en temps d'exécution des requêtes et donc d'obtention des pages ASP.

Concernant les bases de données comme SQL les conseils d'optimisation sont légion et sorte du cadre de ce document, veuillez vous référer aux documents traitant de la question suivant votre moteur.

Concernant le moteur JET (Access), sachez qu'il n'est pas approprié aux bases de données accessibles par un serveur Web. Le modèle concurrentiel et les performances de ce dernier ne sont pas adaptés à un contexte où le nombre d'utilisateurs est potentiellement illimité.
Sachez seulement que, mis à part les considérations communes à toutes les bases de données ('Connection Pooling','System DSN' à la place de 'File DSN', typage des curseurs…), vous pouvez augmenter le nombre de 'locks' dans la base de registre si vous vous heurtez à des échecs dus à de trop nombreux blocages sur une base Access. La valeur qui régit le nombre de locks est *MaxLocksPerFile* (par défaut à 6500) et se trouve sous *HKEY_LOCAL_MACHINE\Software\Microsoft\Jet\4.0\Engines\Jet 4.0*

7. IIS 5.0 et les gains en performance

Avec Windows 2000, arrive entre autre la version 5.0 d'IIS. Nous allons regarder brièvement dans ce chapitre ce qu'apporte cette nouvelle version en termes de performances.

7.1. Nouveautés et améliorations concernant le service WWW

- IIS 4.0 supporte officiellement de 200 à 750 sites web sur le même serveur (chiffre dépendant du type de sites et de leur activité). IIS 5.0 passe cette barre jusqu'à 2000 sites web gérés sur le même serveur.

- Un nouveau modèle d'isolation/protection des applications a été introduit : le "pooled application protection model". C'est le modèle par défaut pour toute nouvelle application ou tout nouveau répertoire virtuel créé dans IIS 5.0.

  Ce paramètre revient à faire un choix entre les applications de confiance et les autres, il est un juste milieu entre les application 'inproc' et les applications tournant en mémoire séparée.
  Avec ce nouveau choix, l'idée est de mettre 'inproc' les application "sûres" et qui ont besoin d'un maximum de performances, de mettre 'out of proc' celles qui sont peu stables et de laisser en 'pooled' les autres applications.

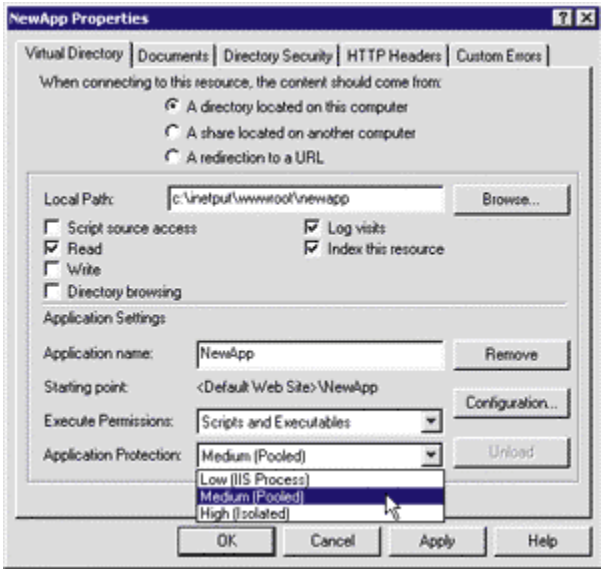


Figure 23 : Un nouveau type d'isolation, "Pooled"

- Une nouvelle propriété dans la metabase "CpuLimitProcStop" permet de spécifier le pourcentage de CPU que les applications pourront utiliser au maximum en un espace de temps donné ("CpuResetInterval"). Par défaut la durée est de 24 heures.

"CpuLimitSop" peut, elle, être définie par l'interface ("CpuResetInterval" ne peut se modifier qu'en modifiant sa valeur dans la metabase)

Exemple: Si cette valeur est fixée à 10%, le processus peut donc utiliser le CPU pour 2.4 heures par jour. A 20% cela fera 4.8 heures.

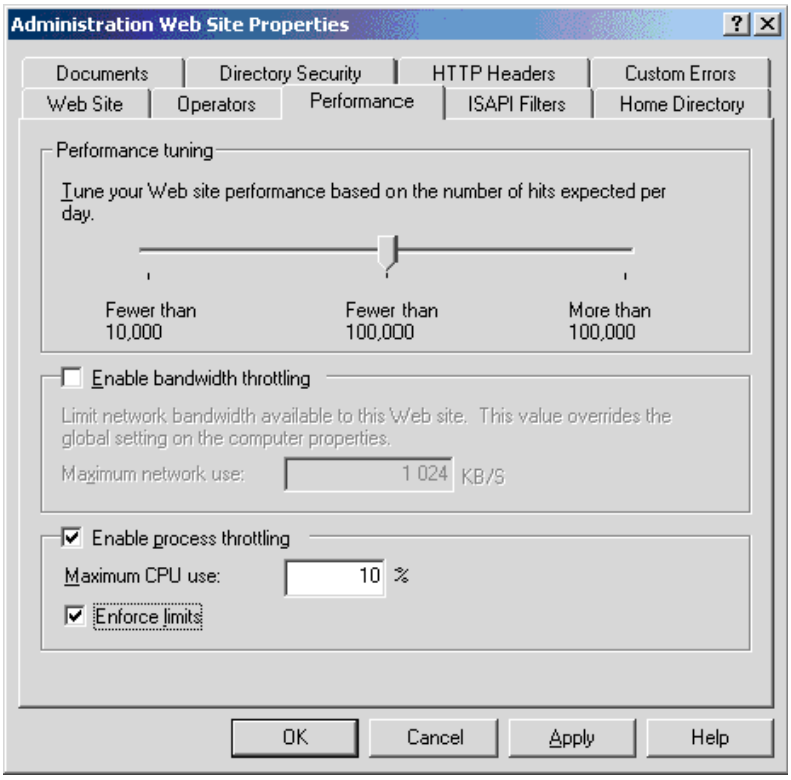


Figure 24 : Fixer des limites de l'utilisation processeur par vos applications

L'option 'Enforce Limit' signifie que la limite ne doit pas s'accompagner que d'avertissements et qu'IIS peut aller jusqu'à arrêter le processus

- Si 100% du temps CPU défini par l'option ci-dessus est atteint (2.4 heures ou 4.8 comme vu au-dessus), IIS ne fait qu'enregistrer cela dans le journal.
- Si 150 % du temps CPU est utilisé: IIS écrit un nouvel événement, puis baisse la priorité du thread associé à l'application.
- Si 200% du temps CPU est utilisé: IIS écrit un nouvel événement puis stoppe le processus: le message "not available" sera alors envoyé au navigateurs.

Ces trois limites se retrouvent dans la metabase:
CpuLimitLogEvent, CpuLimitPriority, et CpuLimitPause

- L'utilisation de Kerberos vous permet de diminuer les baisses de performances liées à la sécurité et à l'authentification. Il garantit un niveau de sécurité supérieur à NTLM tout en déchargeant le serveur web du coût de la gestion de cette sécurité.

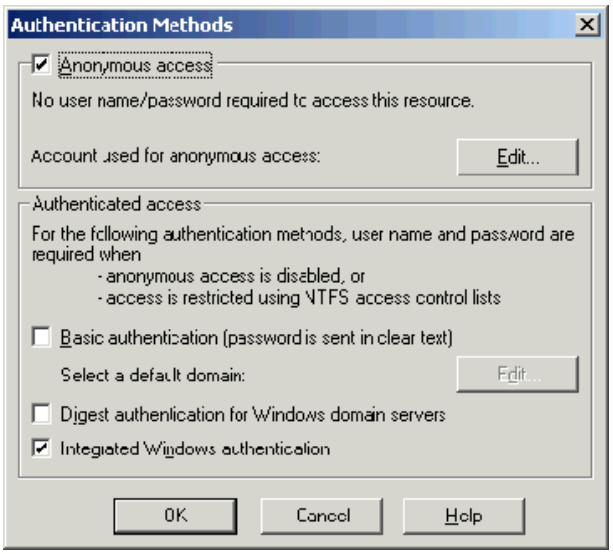


Figure 25 : Kerberos, le mécanisme de sécurité à employer en priorité dans un intranet

7.2. Nouveautés et améliorations concernant ASP

- L'objet Serveur a deux nouvelles méthodes: Server.Transfer et Server.Execute. Ceci permet d'éviter l'aller/retour des Response.Redirect.

- La nouvelle méthode Server.GetLastError renvoi le type d'erreur ou la ligne qui en est à l'origine d'un erreur asp et permet ainsi un debugging de pages plus rapide et facile.

- la nouvelle propriété Response.IsClientConnected permet de savoir si un client est toujours connecté avant de calculer ou envoyer une page. Cela permet des économies en annulant une action sachant que le client n'attend plus de résultats.

- Gestion dynamique du nombre de threads. ASP peut détecter si une ressource externe est à l'origine d'un blocage de requêtes et augmentera alors son nombre de threads. Si le processeur vient a être engorgé, l'inverse se produira pour réduire les 'context switching'.

- Les pages .asp (avec l'extension) sans code ASP sont détectées et le buffering est activé par défaut.

- Tout objet COM qui n'a plus de références et n'utilisant pas la méthode OnEndPage est libéré automatiquement . (Sur IIS 4.0, les objets COM n'étaient libérés que lorsque ASP avait fini de traiter une page)

- Les objets COM de type "Both" qui ne supportent pas le marshaller "Free-Threaded" ne peuvent être stockés dans une variable application (limitation pour éviter les problèmes apparentés à ceux décrits dans le paragraphe 6.9)

- Et d'autres: détermination plus détaillée des possibilités d'un navigateur , passage de "ProcessorThreadMax" et de "ErrorsToNTLog" dans la metabase, encryption des scripts, meilleure gestion des transactions, ...

Conclusion

Pour optimiser votre serveur web, il faut garder à l'esprit les règles suivantes:

- La majorité de l'optimisation doit se faire au niveau de l'application, de son code et des accès aux ressources dont elle dépend.

- Il n'y a pas de valeurs standards pour les différents paramètres existant dans la base de registre ou la metabase: tout dépend de votre configuration logicielle et matérielle, de la charge et de vos choix en terme de priorité (nombre d'utilisateurs par rapport à la vitesse, utilisation mémoire par rapport à utilisation processeur,…).

- Chaque ajustement doit s'accompagner d'une période de tests pour en évaluer les effets et les suites à y donner.

- Performance monitor est l'outil de référence. NetMon sert surtout pour les analyses de trafic et de temps de réponse.

Bibliographie

Les documents suivants ont été utilisés lors de l'élaboration de ce document, vous les trouverez en intégralité dans le MSDN.

Titre	**Auteur** (Microsoft)
100 ways to optimize website performance	Brandon Ballheim
15 ASP Tips to Improve Performance and Style	Nancy Cluts
ASP and Web Session Management	Michael P. Levy
Got Any Cache?	Nancy Winnick Cluts
IIS 4.0 Tuning Parameters for High-Volume Sites	Michael Stephenson
Improving the Performance of Data Access Components with IIS 4.0	Leland Ahlbeck
Managing Session State in a Web Farm	Dennis Angeline
Server Performance and Scalability Killers	George V. Reilly
Tuning Internet Information Server Performance	Mike Moore
Writing High-Performance Active Server Pages	Hans Hugli

Les autres sources sont:
Kit de ressources Techniques IIS
IIS Training Kit
Et les fiches techniques du TechNet…