Benchmarking HOWTO Linux

par André D. Balsa, *andrewbalsa@usa.net* `mailto:andrewbalsa@usa.net`
traduit par François Laagel, *f.laagel@ieee.org* `mailto:f.laagel@ieee.org` v0.12, 15 Août 1997 (v.f. : 2 du 28 Novembre 1997)

Le benchmarking HOWTO Linux traite de certains problèmes relatifs à l'évaluation de performances de systèmes Linux et présente un ensemble d'outils ainsi qu'un formulaire associé qui permettent de produire des mesures de performances significatives en quelques heures. Peut-être ce document contribuera-t-il à une diminution du nombre d'articles inutiles dans comp.os.linux.hardware...

Contents

1 Introduction

"Ce dont on ne peut parler doit être passé sous silence."

Ludwig Wittgenstein (1889-1951), philosophe Autrichien

L'évaluation de performances (benchmarking) consiste à **mesurer** la vitesse à laquelle un ordinateur exécute une tâche calculatoire, et ce de façon à pouvoir comparer différentes configurations logicielles/matérielles. Ceci n'a **aucun** rapport avec la facilité d'utilisation, l'esthétique, les considérations d'ergonomie ou toute autre appréciation subjective.

L'évaluation de performances est une tâche fastidieuse et répétitive. Elle nécéssite que l'on prête une grande attention aux détails. Très souvent les résultats obtenus ne sont pas ceux auxquels on s'attendait et sont sujet à interprétation (ce qui peut très bien être le but d'une procédure d'évaluation de performances).

Enfin, l'évaluation de performances traîte de faits et de chiffres et non pas d'opinion ou d'approximation.

1.1 Pourquoi l'évaluation de performances est-elle si importante ?

Hormis les raisons mentionnées dans le BogoMips Mini-HOWTO (section 7, paragraphe 2), il arrive, lorsque l'on se constitue une machine Linux, que l'on soit confronté à un budget limité et/ou à des besoins en performances minimales garanties.

En d'autres termes, lorsque l'on se pose les questions suivantes :

- Comment maximiser la performance avec un budget donné ?
- Comment minimiser le coût nécéssaire pour obtenir un niveau de performance donné ?
- Comment obtenir le meilleur rapport performance/coût (étant donné un budget ou des besoins en performances minimales garanties) ?

il faudra examiner, comparer et/ou produire des benchmarks (ndt : un benchmark est un programme ou un ensemble de programmes - on parle alors de suite - servant à évaluer les performances d'un système informatique).

Minimiser les coûts sans contraintes de performance implique d'ordinaire la constitution d'une machine à partir de composants de récupération (ce vieux 386SX-16 qui traîne dans le garage sera parfait), et ne nécéssite pas de benchmarks. Maximiser la performance sans coût plafond n'est pas une situation réaliste (à moins que l'on souhaite mettre un Cray dans son salon - la banquette recouverte de cuir qui se trouve au dessus des alimentations électriques est du meilleur goût, n'est-t-il pas ?).

L'évaluation de performances sans contrainte de coût ni de performance minimale garantie n'a pas de sens: c'est une perte de temps et d'argent. L'évaluation de performances n'a de sens que dans le cadre d'une prise de décision, c'est à dire si l'on a le choix entre deux alternatives ou plus.

D'ordinaire des critères autres que le **coût** interviennent dans le processus décisionnel. Il peut s'agir de la disponibilité, du service, de la fiabilité, de considérations stratégiques ou de toute autre caractéristique rationnelle et mesurable d'un système informatique. Par exemple, lorsque l'on compare la performance de différentes versions du noyau Linux, la **stabilité** est toujours plus importante que la vitesse d'exécution.

1.2 Non-critères en matière d'évaluation de performances

Malheureusement et très souvent dans les newsgroups (forums) et les mailing lists (listes de diffusion par courrier électronique), sont cités :

1. La réputation du fabriquant (non-mesurable et sans signification).
2. Les parts de marché du fabriquant (sans signification et non-pertinent).
3. Des paramètres irrationnels (superstition ou a-priori par exemple acheteriez-vous un processeur étiqueté 131313ZAP et peint en rose ?).
4. La valeur perçue (non-significative, non-mesurable et irrationnelle).
5. L'ampleur du tapage marketing (ndt : mercatique pour les intégristes :) est ce qu'il y a de pire, je crois. Personnellement, j'en ai marre des logos "XXX inside" ou "kkkkkws compatible" (maintenant "aaaaaPowered" est de la partie, et puis quoi encore ?). AMHA, les milliards de dollards dépensés durant de telles campagnes seraient bien mieux utilisés par de équipes de recherche pour la conception de nouveaux processeurs, plus rapides (moins chers :-) et moins buggés. Aucune campagne publicitaire, si ambitieuse soit-elle, n'est en mesure de supprimer une bug de la FPU en calcul flottant sur le tout nouveau processeur que vous venez tout juste d'enficher sur votre carte-mère, alors qu'un échange au profit d'un processeur re-conçu le fera.
6. Les opinions du type "Vous avez ce pour quoi vous avez payé" ne sont précisément que ça : des opinions. Donnez-moi des faits, s'il vous plait.

2 Procédures d'évaluation de performances et interprétation des résultats

Quelques recommendations semi-évidentes :

1. Premièrement et avant tout, **identifiez vos objectifs d'évaluation de performances**. Qu'essayez vous exactement d'évaluer ? En quoi un processus d'évaluation de performances va-t-il vous aider à

prendre une décision ultérieure ? Combien de temps et quelles ressources voulez-vous consacrer à cet effort ?

2. **Utilisez des outils standard**. Utilisez une version à jour et stable du noyau, des versions standard et à jour de gcc et de la libc, et un benchmark standard. En bref, utilisez le LBT (voir plus loin).
3. Donnez une **description complète** de votre configuration matérielle (voir le formulaire de compte-rendu plus loin).
4. Essayez d'**isoler une variable unique**. L'évaluation de performances comparative est plus informative que l'évaluation de performances "absolue". **Je n'insisterai jamais assez là-dessus**.
5. **Vérifiez vos résultats**. Faites tourner vos benchmarks plusieurs fois et vérifiez les variations des résultats obtenus, si variation il y a. Des variations inexpliquées invalideront vos résultats.
6. Si vous pensez que votre effort d'évaluation de performances a produit de l'information significative, **partagez-la** avec la communauté Linux de façon **précise** et **concise**.
7. **Oubliez les BogoMips** s'il vous plait. Je me promets d'implémenter un jour un ASIC (ndt : un acronyme pour Application Specific Integrated Circuit, c'est un circuit intégré dédié à une application donnée) dans lequel les BogoMips seront cablés. Et alors on verra ce qu'on verra !

2.1 Comprendre les choix en matière de benchmarks

2.1.1 Benchmarks synthétiques vs. benchmarks applicatifs

Avant de consacrer le moindre temps aux travaux d'évaluation de performances, il importe de faire un choix de base entre benchmarks synthétiques et benchmarks applicatifs.

Les benchmarks synthétiques sont spécifiquement conçus pour mesurer la performance des composants individuels d'un ordinateur, d'habitude en poussant l'un desdits composants jusqu'à sa limite. Un exemple de benchmark synthétique célebre est la suite **Whetstone**, initialement programmée en 1972 par Harold Curnow en FORTRAN (ou était-ce en ALGOL ?), et dont l'usage est toujours très répandu de nos jours. La suite Whetstone produira une mesure de la performance d'une CPU en matière de calcul flottant.

La principale critique que l'on puisse faire aux benchmarks synthétiques est qu'ils ne représentent pas la performance d'un ordinateur, en tant que système complexe, dans des conditions d'utilisation réelles. Prenons par exemple la suite Whetstone, dont la boucle principale est très courte, et qui donc peut aisément tenir dans le cache primaire d'une CPU, cette suite maintient le pipeline de la FPU alimenté en permanence de façon à pousser celle-ci à sa vitesse maximale. On ne peut pas vraiment critiquer la suite Whetstone si l'on se souvient qu'elle a été programmée il y a 25 ans (sa conception est même plus ancienne que ça !), mais il nous faut nous assurer que nous interprétons ses résultats avec prudence quand nous nous préoccupons d'évaluer les performances de micro-processeurs modernes.

Un autre aspect très important qu'il nous faut avoir en tête à propos des benchmarks synthétiques est qu'idéalement, ils devraient pouvoir nous dire quelque chose en ce qui concerne un aspect **spécifique** du système que l'on est en train de tester, et ce indépendamment des autres aspects dudit sysème : un benchmark synthétique d'une carte D'E/S Ethernet devrait produire les mêmes résultats (ou des résultats comparables) que ce soit sur un 386SX-16 avec 4 MB de RAM ou sur un Pentium 200 MMX avec 64 MB de RAM. Si tel n'était pas le cas, le test mesurerait la performance globale de l'association CPU/carte-mère/bus/carte Ethernet/sous-système mémoire/DMA, ce qui n'est pas très utile puisque la différence au niveau des CPUs aura un impact plus important que la différence au niveau des cartes Ethernet (ceci suppose bien sûr que nous utilisions la même combinaison noyau/driver (pilote de périphérique)). Dans le cas contraire la différence de performances pourrait être encore plus grande) !

Enfin, une erreur fréquemment commise est de calculer la moyenne de divers benchmarks synthétiques et de prétendre qu'une telle moyenne est une bonne représentation de la performance globale d'un système donné pour une utilisation dans la vie réelle.

Voici un commentaire sur les benchmarks FPU (cité avec la permission du site Web de Cyrix Corp.) :

> *"Une unité de calcul flottant accélère le logiciel conçu pour l'utilisation de l'arithmétique flottante : typiquement il s'agit de programmes de CAO, de tableurs, de jeux en 3D et d'applications de conception. Cependant, la plupart des applications PC populaires d'aujourd'hui utilisent à la fois des instructions flottantes et l'arithmétique entière. C'est pourquoi Cyrix a choisi de mettre l'accent sur le parallélisme lors de la conception du processeur 6x86 et ce dans le but d'accélérer les programmes qui entremêlent ces 2 types d'instructions.*
>
> *Le modèle de traitement des exceptions flottantes de l'architecture x86 permet aux instructions entières d'être émises et de se terminer pendant qu'une instruction flottante est en cours d'exécution. A l'opposé, une seconde opération flottante ne pourra pas être exécutée alors qu'une précedente instruction flottante est en cours d'exécution. Pour supprimer cette limitation de performance créee par le modèle de traitement des exceptions flottantes, le 6x86, peut émettre spéculativement jusqu'à 4 instructions flottantes vers la FPU intégrée sur le circuit. Par exemple, dans une séquence de code constituée de 2 instructions flottantes (FLTs) suivies de 6 instructions entières (INTs), elles-mêmes suivies de 2 FLTs, le 6x86 peut émettre toutes ces 10 instructions vers les unités d'exécution appropriées avant que l'exécution de la première FLT ne se soit terminée. Si aucune de ces instructions ne provoque d'exception (ce qui est typique), l'exécution continue, les unités flottantes et entières terminant l'exécution de ces instructions en parallèle. Si l'une des FLTs génère une exception (le cas atypique), les possibilités d'exécution spéculatives du 6x86 permettent que l'état du processeur soit restitué de façon à ce que celui-ci soit compatible avec le modèle de traitement des exceptions flottantes.*
>
> *L'examen de code de benchmarks synthétiques flottants révèle que ceux-ci utilisent des séquences d'instructions purement flottantes que l'on ne trouve pas dans les applications du monde réel. Ce type de benchmarks ne tire pas profit des possibilités d'exécution spéculative du processeur 6x86. Cyrix pense que les benchmarks non-synthétiques basés sur des applications du monde réel reflètent mieux la performance que les utilisateurs vont effectivement obtenir. Les applications du monde réel contiennent des instructions entières et flottantes entremêlées et pour cette raison tireront un meilleur parti des possibilités d'exécution spéculative du 6x86."*

La tendance récente en matière d'évaluation de performances consiste donc à choisir des applications usuelles et à les utiliser pour mesurer la performance d'ordinateurs en tant que systèmes complexes. Par exemple **SPEC**, l'entreprise à but non-lucratif qui a conçu les célèbres suites de benchmarks synthétiques SPECINT et SPECFP, a lancé un projet pour développer une nouvelle suite de benchmarks applicatifs. Mais, là encore, il est très improbable qu'une telle suite de benchmarks commerciale comporte du code Linux un jour.

En résumé, les benchmarks synthétiques sont valables à condition d'avoir compris leurs objectifs et leurs limites. Les benchmarks applicatifs reflèteront mieux la performance d'un système informatique, mais aucun d'entre eux n'est disponible pour Linux.

2.1.2 Benchmarks de haut-niveau vs. de bas-niveau

Les benchmarks de bas-niveau ont pour ambition la mesure de la performance du matériel : la fréquence de l'horloge du processeur, les temps de cycle de la DRAM (ndt : acronyme pour Dynamic Random Access Memory) et de la SRAM (ndt : acronyme pour Static Random Access Memory) cache, temps d'accès moyen d'un disque dur, temps d'accès piste à piste, etc...

Cette approche peut être utile si vous avez acheté un système et que vous vous demandez à partir de quels composants il a été construit, bien qu'une meilleure façon de répondre à cette question soit d'ouvrir le boîtier, de dresser l'inventaire des composants que vous trouverez à l'intérieur, et d'obtenir les spécifications techniques pour chacun d'entre eux (elles sont la plupart du temps disponibles sur le Web).

Une autre utilisation possible des benchmarks de bas-niveau consiste à s'en servir pour vérifier qu'un driver du noyau a été correctement configuré pour un composant matériel donné : si vous disposez des spécifications techniques de ce composant vous pourrez comparer les résultats d'un benchmark de bas-niveau aux valeurs théoriques figurant dans les specs.

Les benchmarks de haut-niveau ont plutôt pour objectif la mesure de la performance de l'association matériel/driver/système d'exploitation en ce qui concerne un aspect spécifique d'un système informatique (par exemple la performance des entrées-sorties), ou même une association spécifique matériel/driver/système d'exploitation/application (par exemple un benchmark Apache sur différents ordinateurs).

Bien sûr, tous les benchmarks de bas-niveau sont synthétiques. Les benchmarks de haut-niveau peuvent être synthétiques ou applicatifs.

2.2 Benchmarks standard disponibles pour Linux

AMHA, un test simple que tout le monde peut effectuer à l'occasion d'une mise à jour de la configuration de sa machine Linux est de lancer une compilation du noyau avant et après cette mise à jour matérielle/logicielle, et de comparer les durées de compilation. Si tous les autres paramètres sont les mêmes, alors ce test est valable en tant que mesure de la performance en matière de compilation, et l'on peut affirmer en toute confiance que :

> "Le remplacement de A par B a conduit à une amélioration de x % de la durée de compilation du noyau Linux dans telles et telles conditions".

Ni plus, ni moins !

Parce que la compilation du noyau est une tâche très courante sous Linux, et parce qu'elle met en oeuvre la plupart des fonctionnalités impliquées dans les benchmarks usuels (sauf le calcul flottant), elle constitue un test **isolé** plutôt bon. Cependant, dans la majeure partie des cas, les résultats de ce test ne peuvent pas être reproduits par d'autres utilisateurs de Linux à cause des différences de configurations matérielles/logicielles. Ce test ne constitue donc en aucun cas une métrique permettant de comparer des systèmes dissemblables (à moins que nous ne nous mettions tous d'accord sur la compilation d'un noyau standard - voir plus loin).

Malheureusement, il n'y a pas d'outils d'évaluation de performances ciblant spécifiquement Linux, sauf peut-être les Byte Linux Benchmarks. Ceux-ci sont une version légerement modifiée des Byte Unix Benchmarks qui datent de 1991 (modifications Linux par Jon Tombs, auteurs originels : Ben Smith, Rick Grehan et Tom Yager).

Il existe un *site Web* central pour les Byte Linux Benchmarks.

Une version améliorée et mise à jour des Byte Unix Benchmarks a été synthétisée par David C. Niemi. Elle s'appelle UnixBench 4.01 pour éviter une confusion possible avec des versions antérieures. Voici ce que David a écrit au sujet de ses modifications :

> "Les BYTE Unix benchmarks originels et légerement modifiés sont nases à bien des égards ce qui fait d'eux un indicateur inhabituellement non-fiable de la performance d'un système. J'ai délibérément fait en sorte que mes indices de performance soient très différents pour éviter la confusion avec les vieux benchmarks."

David a mis en place une liste majordomo de diffusion par courrier électronique pour les discussions relatives à l'évaluation de performances sous Linux et sous les systèmes d'exploitation concurrents. Vous pouvez vous joindre à ces discussions en envoyant un e-mail dont le corps contiendra "subscribe bench" à l'adresse

majordomo@wauug.erols.com `mailto:majordomo@wauug.erols.com` . Les groupe des utilisateurs de la région de Washington est aussi en train de mettre en place un *site Web*

concernant les benchmarks sous Linux.

Récemment, Uwe F. Mayer, *mayer@math.vanderbilt.edu* `mailto:mayer@math.vanderbilt.edu` a également porté la suite Bytemark de BYTE sous Linux. Il s'agit d'une suite moderne et compilée très habilement par Rick Grehan du magazine BYTE. Bytemark teste les performances de la CPU, de la FPU et du sous-système mémoire des micro-ordinateurs modernes (ces benchmarks sont strictement orientés vers la performance du processeur, les E/S ou la performance globale du système ne sont pas pris en compte).

Uwe a aussi mis en place un *site Web* , site où l'on peut accéder à une base de données contenant les résultats de sa version des benchmarks BYTEmark pour Linux.

Si vous êtes à la recherche de benchmarks synthétiques pour Linux, vous remarquerez assez vite que sunsite.unc.edu ne propose que peu d'outils d'évaluation de performances. Pour mesurer les performances relatives de serveurs X, la suite xbench-0.2 de Claus Gittinger est disponible sur sunsite.unc.edu, ftp.x.org et d'autres sites (ndt : notamment ftp.lip6.fr qui est l'un des mirroirs de sunsite). Dans son immense sagesse, Xfree86.org refuse de promouvoir ou de recommender le moindre benchmark.

XFree86-benchmarks Survey

est un site Web comprenant une base de données de résultats relatifs à x-bench.

En ce qui concerne les E/S purement disque, l'utilitaire hdparm (qui fait partie de la plupart des distributions, mais est aussi disponible sur sunsite.unc.edu) permet de mesurer les taux de transfert grâce aux options -t et -T.

Il existe plein d'autres outils disponibles librement (sous license GPL) sur Internet pour tester divers aspects de la performance de votre machine Linux.

2.3 Liens et références

La FAQ du newsgroup comp.benchmarks par Dave Sill est la référence standard en matière d'évaluation de performances. Elle n'est pas particulièrement orientée Linux, mais elle n'en reste pas moins une lecture recommendée pour tous ceux qui font preuve d'un minimum de sérieux envers le sujet. Elle est disponible sur nombre de sites FTP et de sites Web et recense **56 benchmarks différents** avec des liens vers des sites FTP permettant de les télécharger. Cependant, certains des benchmarks recensés sont des produits commerciaux.

Je n'entrerai pas dans la description détaillée des benchmarks mentionnés dans la FAQ de comp.benchmarks, mais il y a au moins une suite de bas-niveau au sujet de laquelle j'aimerai faire quelques commentaires : la *suite lmbench* de Larry McVoy. Je cite David C. Niemi :

> *"Linus et David Miller s'en servent beaucoup parce qu'elle permet des mesures de bas-niveau utiles et peut aussi quantifier la bande passante et la latence d'un réseau si vous avez deux machines à votre disposition pour le faire tourner. Mais lmbench n'essaie pas de produire un indice de performance global..."*

Un *site FTP*

assez complet en matière de benchmarks disponibles **librement** a été mis en place par Alfred Aburto. La suite Whetstone utilisée dans le LBT est disponible sur ce site.

Une **FAQ multi-fichier par Eugene Miya** est également postée sur comp.benchmarks; c'est une excellente référence.

3 Le Linux Benchmarking Toolkit (LBT)

Je propose ici un ensemble d'outils pour l'évaluation de performances sous Linux. C'est la version préliminaire d'un vaste environnement d'évaluation de performances pour Linux, il est destiné à être amélioré et à voir ses fonctionnalités étendues. Prenez le pour ce qu'il vaut, c'est-à-dire une proposition. Si vous pensez que cette suite de test n'est pas valable, prenez la liberté de m'envoyer (ndt : à l'auteur et non au traducteur, merci :-) vos critiques par e-mail et soyez sûrs que je serai heureux d'intégrer les changements que vous aurez suggéré dans la mesure du possible. Avant d'entamer une polémique, lisez ce HOWTO et les documents cités en référence : les critiques informés sont les bienvenus, les critiques stériles ne le sont pas.

3.1 Motivations

Elles sont dictées par le bon sens, tout simplement :

1. Cette suite ne doit pas nécessiter plus d'une journée de durée d'exécution. En matière de benchmarks comparatifs (diverses exécutions), personne ne veut passer des jours à essayer de trouver la configuration matérielle le plus rapide pour un système donné. Idéalement, l'ensemble de la suite devrait pouvoir tourner en 15 minutes sur une machine moyenne.
2. Tout le code source doit être disponible librement sur le Net, pour des raisons évidentes.
3. Les benchmarks devraient fournir des chiffres simples et reflétant la performance mesurée.
4. Il devait y avoir un mélange de benchmarks synthétiques et de benchmarks applicatifs.
5. Chacun des benchmarks **synthétiques** devrait pousser un sous-système particulier à ses limites.
6. Les résultats des benchmarks **synthétiques** ne devraient **pas** être combinés par le biais d'une moyenne afin d'en extraire un facteur de mérite global (cela va à l'encontre du principe fondateur des benchmarks synthétiques et conduit à une perte d'information considérable).
7. Les benchmarks applicatifs devraient être représentatifs de tâches couramment exécutées sur des systèmes Linux.

3.2 Sélection des benchmarks

J'ai sélectionné 5 suites des benchmarks différentes en évitant autant que possible les recouvrements dans les tests :

1. Compilation du noyau 2.0.0 (configuration par défaut) avec gcc.
2. Whetstone version 10/03/97 (la version la plus récente de Roy Longbottom).
3. xbench-0.2 (avec les paramètres d'exécution rapide).
4. Les benchmarks UnixBench version 4.01 (résultats partiels).
5. Les benchmarks de la suite BYTEmark du magazine BYTE beta release 2 (résultats partiels).

Pour les tests 4 et 5, "(résultats partiels)" signifie qu'une partie seulement des résultats produits est prise en compte.

3.3 Durée des tests

1. Compilation du noyau 2.0.0 : 5 - 30 minutes, selon la performance **réelle** de votre machine.
2. Whetstone : 100 secondes.
3. Xbench-0.2 : $<$ 1 heure.
4. Les benchmarks d'UnixBench version 4.01 : environs 15 minutes.
5. Les benchmarks de la suite BYTEmark du magazine BYTE : environs 10 minutes.

3.4 Commentaires

3.4.1 Compilation du noyau 2.0.0

- **Quoi :** c'est le seul benchmark applicatif de la LBT.
- Le code est largement disponible (càd que j'ai finalement trouvé une utilisation pour mes vieux CD-ROMs Linux).
- La plupart des linuxiens recompilent leur noyau assez souvent, c'est donc une mesure significative de la performance globale.
- Le noyau est gros et gcc utilise une bonne partie de la mémoire (ndt : surtout à l'édition de liens) : ceci contribue à atténuer le biais induit par le cache L2 lorsque l'on se contente de passer de petits tests.
- Les E/S vers le disque sont fréquentes.
- Procédure de test : trouvez une antique arborescence source de 2.0.0, compilez la avec les options par défaut (make config, appuyez sur Enter autant de fois que nécéssaire). Le temps affiché doit correspondre à la durée passée sur la compilation càd après que vous ayez tapé make zImage (sans prendre en compte le make dep clean). Notez que l'architecture cible par défaut est i386, donc si vous compilez sur une autre architecture, gcc devrait être en mesure de cross-compiler en utilisant i386 en tant qu'architecture cible.
- **Résultats :** la durée de compilation en minutes et secondes (s'il vous plait, ne rapportez pas les fractions de secondes).

3.4.2 La suite Whetstone

- **Quoi :** mesure la performance en calcul flottant pur à l'intérieur d'une courte (et dense) boucle. Le code source (en C) est assez lisible et il est très facile de voir quelles sont les opérations flottantes impliquées.
- C'est le plus court des tests de la LBT :-).
- C'est un "Vieux Classique" : des chiffres sont disponibles pour comparaison, ses défauts et ses faiblesses sont bien connues.
- Procédure de test : le code source le plus récent devrait être téléchargé depuis le site d'Aburto. Compilez le et exécutez le en mode double précision. Spécifiez gcc et -O2 en tant que pré-processeur et option du compilateur respectivement. Définissez aussi la variable du pré-processeur POSIX à 1 pour préciser le type de machine.
- **Résultats :** un indice de performance en calcul flottant exprimé en MWIPS.

3.4.3 Xbench-0.2

- **Quoi :** mesure la performance de serveurs X.
- La mesure en xStones fournie par xbench est une moyenne pondérée de quelques tests rapportés aux performances obtenues sur une vieille station Sun ne disposant que d'un display d'un seul bit de profondeur (ndt : en clair, c'est du monochrome pur et dur). Mouais... on peut légitimement se demander si xbench est véritablement adéquat en tant que test pour des serveurs X modernes. Néanmoins, c'est le meilleur outil que j'ai trouvé.
- Procédure de test : compilez avec -O2. On spécifiera aussi quelques options pour une exécution courte : `./xbench -timegoal 3 > results/name_of_your_linux_box.out`. Pour générer l'indice xStones, il nous faudra encore lancer un script awk; la façon la plus simple de le faire étant de taper make summary.ms. Jetez un coup d'oeuil au fichier summary.ms : l'indice xStone de votre système est dans la dernière colonne de la ligne contenant le nom de votre machine – nom que vous aurez spécifié pendant le test.
- **Résultats :** un indice de performances exprimé en xStones.
- Note : ce test, en l'état, est dépassé. Il devrait être ré-écrit.

3.4.4 UnixBench version 4.01

- **Quoi** : mesure la performance globale d'un système UNIX. Ce test met en oeuvre évidence la performance des E/S fichier et de la gestion du multi-tâches par le noyau.
- J'ai supprimé tous les résultats de tests arithmétiques et je n'ai conservé que les tests orientés système.
- Procédure de test : make avec -O2. Exécution avec `./Run -1` (tournez chacun des tests une fois). Vous trouverez les résultats dans le fichier ./results/report. Calculez la moyenne géométrique des indices de EXECL THROUGHPUT, FILECOPY 1, 2, 3, PIPE THROUGHPUT, PIPE-BASED CONTEXT SWITCHING, PROCESS CREATION, SHELL SCRIPTS et SYSTEM CALL OVERHEAD.
- **Résultats :** un indice de performance du système.

(ndt : la moyenne géométrique se définit comme la racine n-ième du produit des n termes considérés)

3.4.5 Les benchmarks Bytemark du magazine BYTE

- Quoi : fournit une bonne mesure de la performance CPU. Voici un extrait de la documentation : *"Ces benchmarks ont étés conçu de façon à mettre en évidence la limite supérieure de la CPU, de la FPU et de l'architecture mémoire d'un système. Ils ne peuvent mesurer la performance du sous-système graphique, des accès disque ou du débit réseau (ce sont là le domaine d'un ensemble différent de benchmarks). C'est pourquoi, il est souhaitable que vous n'utilisiez les résultats de ces tests qu'en tant qu'élément d'appréciation partielle, et non pas totale, lors de l'évaluation de la performance d'un système."*
- J'ai supprimé tous les résultats de test FPU puisque la suite Whetstone est tout aussi représentative des performances à cet égard.
- J'ai décomposé les tests entiers en deux groupes : ceux qui sont plus représentatifs de la performance cache mémoire/CPU, et ceux qui utilisent l'arithmétique entière de la CPU.

- Procédure de test : make avec -O2. Exécutez le test avec ./nbench >myresults.dat ou quelque chose d'approchant. Puis, à partir de myresults.dat, calculez la moyenne géométrique des indices des tests STRING SORT, ASSIGNMENT et BITFIELD. Ceci vous donnera l'**indice mémoire**. Calculez la moyenne géométrique des indices des tests NUMERIC SORT, IDEA, HUFFMAN et FP EMULATION: c'est l'**indice entier**.
- **Résultats :** un indice mémoire et un indice entier calculés comme expliqué ci-dessus.

3.5 Améliorations possibles

La suite de benchmarks idéale tournerait en quelques minutes, comprendrait des benchmarks synthétiques testant chaque sous-système séparément et des benchmarks applicatifs fournissant des résultats pour différentes applications. Elle produirait aussi automatiquement un rapport complet et éventuellement l'enverrait par e-mail à une base de données centrale sur le Web.

La portabilité n'est pas vraiment notre souci premier dans cette affaire. Pourtant, une telle suite devrait tourner au moins sur toutes les versions (> 2.0.0) du noyau Linux, et ce dans toutes leurs déclinaisons possibles (i386, Alpha, Sparc...).

Si quelqu'un a la moindre idée concernant l'évaluation de performances réseau au moyen d'un test à la fois simple, facile d'emploi, fiable, et dont la mise en oeuvre prendrait moins de 30 minutes (configuration et exécution), s'il vous plait contactez-moi.

3.6 Formulaire de rapport LBT

Au-delà des tests, la procédure d'évaluation de performances n'aurait pas été complète sans un formulaire décrivant la configuration matérielle utilisée lors de leur exécution. Le voici donc : (il se conforme aux directives prescrites dans la FAQ de comp.benchmarks) :

(ndt : le formulaire en question n'a délibérément pas été traduit, de façon à ce que l'auteur de ce HOWTO puisse automatiser leur traitement en vue de maintenir une base de données de résultats. Voir la section 4 pour un exemple de formulaire correctement rempli).

```
______________________________________________________________________
LINUX BENCHMARKING TOOLKIT REPORT FORM
______________________________________________________________________

______________________________________________________________________
CPU
==
Vendor:
Model:
Core clock:
Motherboard vendor:
Mbd. model:
Mbd. chipset:
Bus type:
Bus clock:
Cache total:
Cache type/speed:
SMP (number of processors):
______________________________________________________________________

______________________________________________________________________
```

```
RAM
====
Total:
Type:
Speed:
-----------------------------------------------------------------------

-----------------------------------------------------------------------
Disk
====
Vendor:
Model:
Size:
Interface:
Driver/Settings:
-----------------------------------------------------------------------

-----------------------------------------------------------------------
Video board
===========
Vendor:
Model:
Bus:
Video RAM type:
Video RAM total:
X server vendor:
X server version:
X server chipset choice:
Resolution/vert. refresh rate:
Color depth:
-----------------------------------------------------------------------

-----------------------------------------------------------------------
Kernel
=====
Version:
Swap size:
-----------------------------------------------------------------------

-----------------------------------------------------------------------
gcc
===
Version:
Options:
libc version:
-----------------------------------------------------------------------

-----------------------------------------------------------------------
Test notes
==========
-----------------------------------------------------------------------

-----------------------------------------------------------------------
RESULTS
========
```

```
Linux kernel 2.0.0 Compilation Time: (minutes and seconds)
Whetstones: results are in MWIPS.
Xbench: results are in xstones.
Unixbench Benchmarks 4.01 system INDEX:
BYTEmark integer INDEX:
BYTEmark memory INDEX:
-----------------------------------------------------------------------

-----------------------------------------------------------------------
Comments*
=========
* Ce champ n'est présent dans ce formulaire que pour de possibles
interprétations des résultats, et tant que tel, il est optionnel.
Il pourrait cependant constituer la partie la plus importante de votre
compte-rendu, tout particulièrement si vous faites de l'évaluation
de performances comparative.
-----------------------------------------------------------------------
```

3.7 Test de performances réseau

Le test des performances réseau est un véritable défi en soi puisqu'il implique au moins deux machines: un serveur et une machine cliente. Pour cette raison ce genre de test nécessite deux fois plus de temps pour être mis en place, il y a plus de variables à contrôler, etc... Sur un réseau Ethernet, il me semble votre meilleure option est le paquetage ttcp. (à developper)

3.8 Les tests SMP

Les tests SMP sont un autre défi, et tout test conçu spécifiquement pour un environnement SMP aura des difficultés à s'avérer valide dans des conditions d'utilisation réelles parce que les algorithmes qui tirent parti du SMP sont difficiles à développer. Il semble que les versions du noyau Linux les plus récentes (> 2.1.30 ou pas loin) feront du scheduling (ndt : ordonnancement de thread ou de processus) à grain fin ; je n'ai pas plus d'information que ça pour le moment.

Selon David Niemi, *"... shell8* de la suite Unixbench 4.01 *fait du bon travail en matière de comparaison de matériel/SE similaires en mode SMP et en mode UP."*

(ndt : SMP = Symetric Multi-Processing, UP = Uni-Processor)

4 Une exécution type et les résultats

Le LBT a été lancé sur ma machine perso., une machine de classe Pentium tournant Linux que j'ai assemblée moi-même et que j'utilise pour écrire ce HOWTO. Voici le compte-rendu LBT pour ce système :

```
LINUX BENCHMARKING TOOLKIT REPORT FORM

CPU
==

Vendor: Cyrix/IBM

Model: 6x86L P166+
```

```
Core clock: 133 MHz

Motherboard vendor: Elite Computer Systems (ECS)

Mbd. model: P5VX-Be

Mbd. chipset: Intel VX

Bus type: PCI

Bus clock: 33 MHz

Cache total: 256 KB

Cache type/speed: Pipeline burst 6 ns

SMP (number of processors): 1

RAM
====

Total: 32 MB

Type: EDO SIMMs

Speed: 60 ns

Disk
====

Vendor: IBM

Model: IBM-DAQA-33240

Size: 3.2 GB

Interface: EIDE

Driver/Settings: Bus Master DMA mode 2

Video board
===========

Vendor: Generic S3

Model: Trio64-V2

Bus: PCI

Video RAM type: EDO DRAM

Video RAM total: 2 MB

X server vendor: XFree86
```

```
X server version: 3.3

X server chipset choice: S3 accelerated

Resolution/vert. refresh rate: 1152x864 @ 70 Hz

Color depth: 16 bits

Kernel
=====

Version: 2.0.29

Swap size: 64 MB

gcc
===

Version: 2.7.2.1

Options: -O2

libc version: 5.4.23

Test notes
==========

Une charge très faible. Les tests ci-dessus ont été exécutés
avec quelques unes des options spécifiques du Cyrix/IBM 6x86 activées
grâce au programme setx86. Il s'agit de : fast ADS, fast IORT, Enable DTE,
fast LOOP, fast Lin. VidMem.

RESULTS
========

Linux kernel 2.0.0 Compilation Time: 7m12s

Whetstones: 38.169 MWIPS.

Xbench: 97243 xStones.

BYTE Unix Benchmarks 4.01 system INDEX: 58.43

BYTEmark integer INDEX: 1.50

BYTEmark memory INDEX: 2.50

Comments
=========

Il s'agit là d'une configuration très stable et dotée de performances
homogènes, idéale pour une utilisation individuelle et/ou développer
sous Linux. Je rendrai compte des résultats obtenus avec un 6x86MX dès
que j'aurai réussi à mettre la main sur l'un d'entre eux !
```

5 Pièges et mises en garde en matière d'évaluation de performances

Après avoir compilé ce HOWTO, j'ai commencé à comprendre pourquoi les mots de "piège" et de "mise en garde" sont si souvent associés à l'évaluation de performances.

5.1 Comparer des pommes et des oranges

Ou devrais-je dire des Apples et des PCs ? (ndt : pour goûter pleinement toute la finesse de ce jeu de mots, il faut quand même savoir que pomme se dit apple en anglais :-) C'est tellement évident et c'est une controverse tellement éculée que je ne rentrerai pas dans les détails. Je doute que le temps nécessaire pour booter un Mac comparé à celui d'un Pentium moyen soit une véritable mesure de quoi que ce soit. De façon similaire on pourrait parler du boot de Linux vs. Windows NT, etc... Essayez autant que possible de comparer des machines identiques à une seule différence près.

5.2 Information incomplète

Un seul exemple suffira à l'illustration de cette erreur très courante. On lit souvent dans comp.os.linux.hardware l'affirmation suivante : "Je viens tout juste d'enficher le processeur XYZ qui tourne à nnn MHz et la compilation du noyau prend maintenant i minutes (ajustez XYZ, nnn et i selon vos besoins). C'est énervant parce qu'aucune autre information n'est fournie: on ne connait même pas la quantité de RAM, la taille du swap, les autres tâches qui tournaient à ce moment là, la version du noyau, les modules sélectionnés, le type de disque dur, la version de gcc, etc...

Je vous recommende d'utiliser le formulaire de compte-rendu LBT, lequel fournit au moins un cadre informationnel standard.

5.3 Matériel/logiciel propriétaire

Un fabriquant de micro-processeurs bien connu a publié naguère des résultats de benchmarks produits avec une version spéciale et personnalisée de gcc. Considérations éthiques mises à part, ces résultats sont dénués de toute signification, en effet, la totalité de la communauté Linux aurait utilisé la version standard de gcc. Le même genre de considération vaut aussi pour le matériel propriétaire. L'évaluation de performances est beaucoup plus utile quand elle va de pair avec du matériel sur étagère et du logiciel gratuit (au sens GNU/GPL).

5.4 Pertinence

On parle de Linux, non ? On devrait donc faire abstraction des benchmarks produits sous d'autres systèmes d'exploitation (ceci est une instance particulière de la comparaison des pommes et des oranges mentionnée plus haut). Si l'on se propose d'évaluer la performance d'un serveur Web, on pourra aussi se dispenser de citer la performance FPU et toute autre information non-pertinente. Dans de tels cas, moins c'est plus. Enfin, vous n'avez pas non plus besoin de parler de l'age de votre chat, de votre humeur pendant que vous procédez à une évaluation de performances, etc...

6 FAQ

Q1.

Existe-t-il un indice de performances spécifique aux machines Linux ?

A.

Non, Dieu merci personne n'a encore inventé de mesure du type Lhinuxstone (tm). Même si c'était le cas, ça n'aurait pas beaucoup de sens : les machines Linux sont utilisées pour des tâches tellement différentes allant des serveurs Web lourdement chargés aux stations graphiques pour utilisation individelle. Aucun facteur de mérite ne peut décrire les performances d'une machine Linux dans des situations si différentes.

Q2.

Alors, pourquoi ne pas choisir une douzaine de métriques pour résumer la performance de diverses machines Linux ?

A.

Ça serait une situation idéale. J'aimerai voir ça devenir une réalité. Y-a-t-il des volontaires pour un **projet d'évaluation de performances sous Linux** ? Avec un site Web et une base de données de rapports bien conçus, complète et en ligne ?

Q3.

... BogoMips ...

A.

Les BogoMips n'ont strictement rien à voir avec la performance de votre machine. Voir le BogoMips Mini-HOWTO.

Q4.

Quel est le "meilleur" benchmark pour Linux ?

A.

Ça dépend complètement de quel aspect des performances d'une machine Linux on souhaite mesurer. Il y a des benchmarks différents pour faire des mesures réseau (taux de transfert soutenu sous Ethernet), des mesures de serveur de fichiers (NFS), de bande passante, de performance CAO, de temps de transaction, de performance SQL, de performances de serveur Web, de performance temps-réel, de performance CD-ROM, de performance sous Quake (!), etc ... Pour autant que je sache, aucune suite de benchmarks supportant tous ces tests n'existe pour Linux.

Q5.

Quel est le processeur le plus rapide pour Linux ?

A.

Le plus rapide pourquoi faire ? Si on est plutôt orienté calcul intensif, alors un Alpha à fréquence d'horloge élevée (600 MHz et ça continue à grimper) devrait être plus rapide que n'importe quoi d'autre, du fait que les Alphas ont été conçus dans cette optique. D'un autre côté, si vous voulez vous constituer un serveur de news très rapide, il est probable que le choix d'un sous-système disque rapide et de beaucoup de RAM vous menera à de meilleures améliorations de performances qu'un changement de processeur (à prix constant).

Q6.

Laissez-moi reformuler la dernière question, alors : y-a-t-il un processeur qui soit le plus rapide dans les applications d'usage général ?

A.

C'est une question délicate mais la réponse est simple : NON. On peut toujours concevoir un système plus rapide même pour des applications d'usage général indépendamment du processeur. D'ordinaire, en conservant tous les autres paramètres à leur valeur nominale, des fréquences d'horloge plus élevées permettent d'obtenir de meilleures performances (ndt : surtout si on parle de systèmes synchrones :-) et aussi plus de maux de tête. Si vous retirez un vieux Pentium à 100 MHz d'une carte-mère (laquelle n'est pas souvent) upgradable, et que vous le remplaciez par un Pentium 200 MHz MMX vous devriez sentir une différence de performances notable. Bien sûr, pourquoi se contenter de 16 MB de RAM ? Le même investissement aurait été fait de façon encore plus avisée au profit de quelques SIMMs supplémentaires...

Q7.

Donc la fréquence d'horloge du processeur a une influence sur la performance d'un système ?

A.

La plupart des tâches sauf les boucles de NOP (qui sont d'ailleurs supprimées à la compilation par les compilateurs modernes) une augmentation de la fréquence d'horloge permettra d'obtenir une augmentation linéaire de la performance. Des applications gourmandes en ressources CPU et très petites (pouvant donc tenir dans le cache L1 : 8 ou 16KB) verront leur performances augmenter dans la même proportion que l'augmentation de la fréquence d'horloge. Cependant les "vrais" programmes comportent des boucles qui ne tiennent pas dans le cache primaire, doivent partager le cache secondaire (externe) avec d'autres processus et dépendent de composants externes (ndt : pour les E/S) bénéficieront d'un gain de performance beaucoup moins important. Tout ceci parce que le cache L1 fonctionne à la même fréquence d'horloge que le processeur, alors que la plupart des caches L2 et des autres sous-systèmes (DRAM par exemple) tournent en asynchrone à des fréquences plus basses.

Q8.

D'accord, dans ce cas, une dernière question sur le sujet : quel est le processeur présentant le meilleur rapport prix/performance pour une utilisation d'usage général sous Linux ?

A.

Définir une "utilisation d'usage général sous Linux" n'est pas chose facile ! Etant donnée une application quelconque, il y a toujours moyen de déterminer quel processeur du moment détient le meilleur rapport prix/performance pour ladite application. Mais les choses changent si rapidement à mesure que les fabriquants diffusent de nouveaux processeurs, que dire "le processeur XYZ à n MHz est le choix du moment" serait forcément réducteur. Cependant, le prix du processeur n'est pas significatif dans le coût d'un système complet que l'on assemble soi-même. Donc, la question devrait plutôt être "comment maximize-t-on le rapport performance/coût d'une machine donnée ?" Et la réponse à cette question dépend en grande partie des besoins en performance minimale garantie et/ou du coût maximal de la configuration que l'on considère. Il arrive parfois que le matériel sur étagère ne permette pas d'atteindre les besoins en performance minimale garantie que l'on souhaite obtenir et que des systèmes RISC coûteux soient la seule alternative viable. Pour une utilisation personnelle, je recommende une configuration équilibrée et homogène du point de vue de la performance globale (et maintenant débrouillez vous pour deviner ce que j'entends par équilibré et homogène :-); le choix d'un processeur est une décision importante, mais pas plus que celle du disque dur et de sa capacité, celle de la quantité de RAM, celle de la carte graphique, etc...

Q9.

Qu'est-ce qu'une amélioration significative des performances ?

A.

Je dirais que tout ce qui est sous 1% n'est pas significatif (pourrait être décrit comme marginal). Nous autres, simples mortels, avons du mal à percevoir la différence entre 2 systèmes dont les temps de réponses sont distants de moins de 5% . Bien sûr, certains évaluateurs de performances - plutôt de la tendance intégriste - ne sont aucunement humains et vous raconteront, en comparant 2 systèmes dont les indices de performances sont de 65.9 et de 66.5, que ce dernier est indubitablement plus rapide.

Q10.

Comment puis-je obtenir une amélioration significative de performance à moindre coût ?

A.

Puisque le code source complet de Linux est disponible, un examen attentif et une re-conception algorithmique de procédures clés peuvent, dans certains cas, déboucher sur des améliorations jusqu'à un facteur 10 en terme de performance. Si l'on est sur un projet commercial et qu'on ne souhaite pas plonger dans les tréfonds du code source du système, **l'implication d'un consultant Linux est nécéssaire**. Cf. le Consultants-HOWTO.

7 Copyright, remerciements et divers

7.1 Comment ce document a-t-il été produit

La première étape a consisté en la lecture de la section 4 "Writing and submitting a HOWTO" de l'index des HOWTOs écrit par Greg Hankins.

Je ne savais absolument rien au sujet de SGML ou de LaTeX mais, après avoir lu divers commentaires à propos de SGML-Tools, j'étais tenté d'utiliser un paquetage de génération de documentation automatique. Cependant l'insertion manuelle de directives de formattage me rappelait l'époque où j'assemblais à la main un moniteur 512 octets pour un processeur 8 bits aujourd'hui disparu. J'ai donc fini par récupérer les sources de LyX, les compiler et je m'en suis servi dans son mode LinuxDoc. Une association chaudement recommendée : **LyX et SGML-Tools**.

7.2 Copyright

7.3 Nouvelles version de ce document

De nouvelles version du Benchmarking-HOWTO Linux seront mises à disposition sur sunsite.unc.edu et sur les sites mirroir (ndt : citons ftp.lip6.fr pour nous autres francophones). Ce document existe aussi sous d'autres formats tels que PostScript et dvi, et sont disponibles dans le répertoire other-formats. Le Benchmarking-HOWTO Linux est également disponible pour des clients WWW comme Grail, un butineur Web écrit en Python. Il sera aussi posté régulièrement sur comp.os.linux.answers.

7.4 Retour

Les suggestions, corrections, et ajouts sont désirés. Les contributeurs sont les bienvenus et en seront remerciés. Les incendies (ndt : est-ce une traduction acceptable de flame ?) sont à rediriger sur /dev/null.

Je serai toujours joignable à andrewbalsa@usa.net.

7.5 Remerciements

David Niemi, l'auteur de la suite Unixbench, s'est averé être une source inépuisable d'informations et de critiques (fondées).

Je veux aussi remercier Greg Hankins, le coordinateur des HOWTO Linux et l'un des principaux contributeurs au paquetage SGML-tools, Linus Torvalds et toute la communauté Linux. Ce HOWTO est ma façon de renvoyer l'ascenseur.

7.6 Paravent

Votre kilomètrage peut varier et variera sans doutes. Soyez conscients que l'évaluation de performances est un sujet très sensible et une activité qui consomme énormément de temps et d'énergie.

7.7 Marques déposées

Pentium et Windows NT sont des marques déposées d'Intel et de Microsoft Corporations respectivement.

BYTE et BYTEmark sont des marques déposées de McGraw-Hill, Inc.

Cyrix et 6x86 sont des marques déposées de Cyrix Corporation.

Linux n'est pas une marque déposée, et espérons qu'elle ne le sera jamais.