MIF25 : SYSTEMES TEMPS-REEL

TP : SYSTEME GENERALISTE ET TEMPS DE REPONSE

AYNE LORIS 11208025, LIU ZHUYING 11306849

1. EXPERIMENTATION

Objectif : Prouver que Linux n'est pas temps-réel.

Afin de démontrer que les systèmes généralistes ne sont pas temps-réel, nous avons utilisé des fonctions simples appelées de manière répétée sur des périodes de temps plus ou moins grandes. L'idée est de stresser le système afin de voir apparaitre des retards d'exécution, signe d'un système non temps-réel.

Les expériences ont été menées sur un MSI GT70 Dont voici les caractéristiques importantes :

Système d'exploitation	Ubuntu 14.04 LTS - 64 bits
Disque dur	HDD 540 Go
Processeur	Intel Core i7-3630QM - 8 x 2.40GHz
Mémoire	12 Go DDR3

Le programme utilisé est composé de 3 fichiers (main.c, utilities.c et utilities.h) et d'un makefile. Les fichiers « utilities » contiennent toutes les fonctions d'affichage et de traitement des TimeSpec. Le fichier « main » contient quant à lui la lecture des arguments et l'exécution répétée des fonctions passives et actives.

Dans le cas de l'exécution passive, le programme utilise un signal périodique pour exécuter la fonction. Dans le cas actif, le programme surveille l'horloge de façon intensive et lance la fonction après « dt » microsecondes. Dans les deux cas, la fonction est chargée de calculer la différence entre son temps d'exécution réel et son temps d'exécution théorique. Si cette différence est non nulle, alors un retard existe.

Pour stresser le système d'exploitation, nous avons utilisé « Stress », une application Linux permettant de pousser le CPU à 100% sur un nombre choisi de cœurs (cf. Figure 1).

Le nombre de tests a été fixé à 500, nombre suffisant à la création d'un retard sur l'ordinateur, et permettant de gagner du temps, sans quoi les expériences deviennent trop longues pour un résultat similaire (l'objectif est d'obtenir des retards, pas les plus grands retards !) (cf. Figure 2).

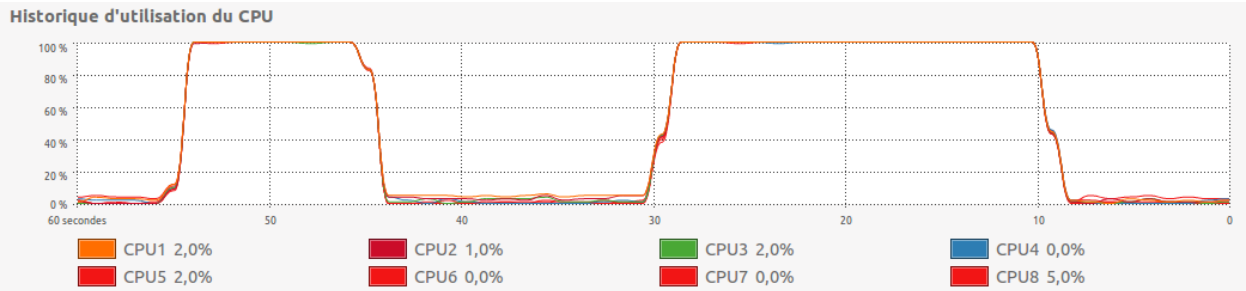


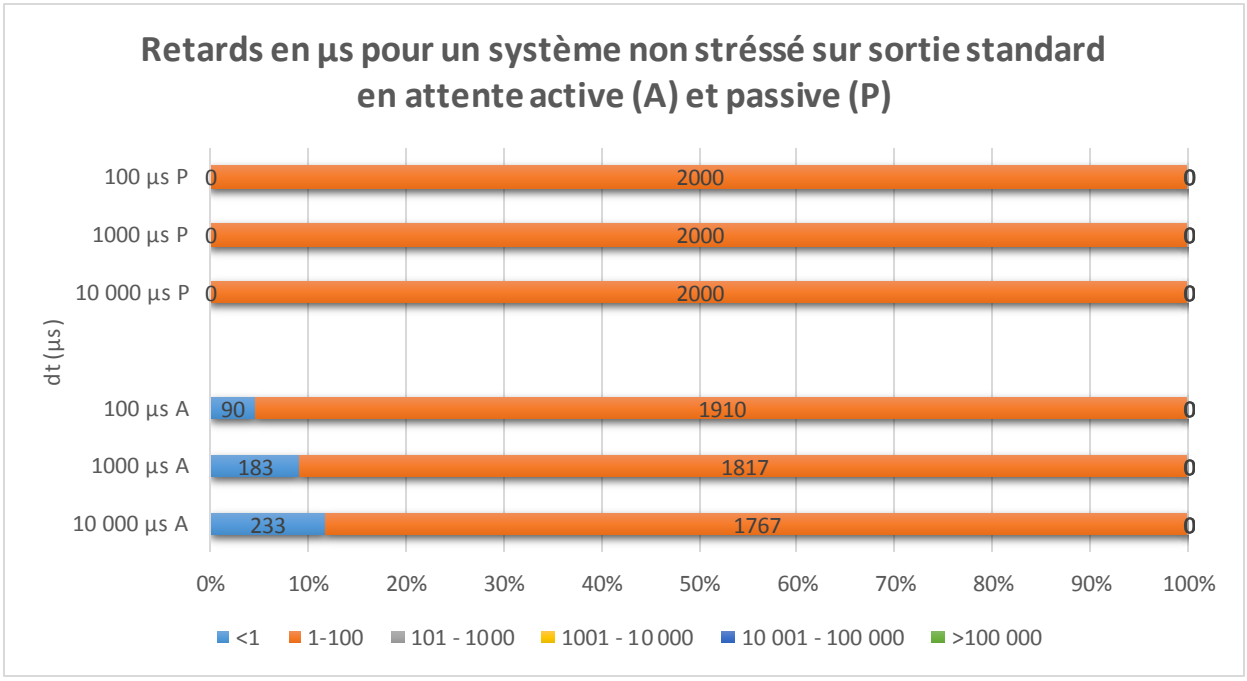
Figure 1 : Effet d'un stress de 10 puis de 20 secondes sur les 8 cœurs.

Expérience	Nombre de tests	Dt (µs)	Attente	OS occupé
1	500	10000	Active	Non
2	500	1000	Active	Non
3	500	100	Active	Non
4	500	10000	Passive	Non
5	500	1000	Passive	Non
6	500	100	Passive	Non
7	500	10000	Active	Oui
8	500	1000	Active	Oui
9	500	100	Active	Oui
10	500	10000	Passive	Oui
11	500	1000	Passive	Oui
12	500	100	Passive	Oui

Figure 2 : Tableau récapitulatif des expériences menées.

2. RESULTATS ET ANALYSE

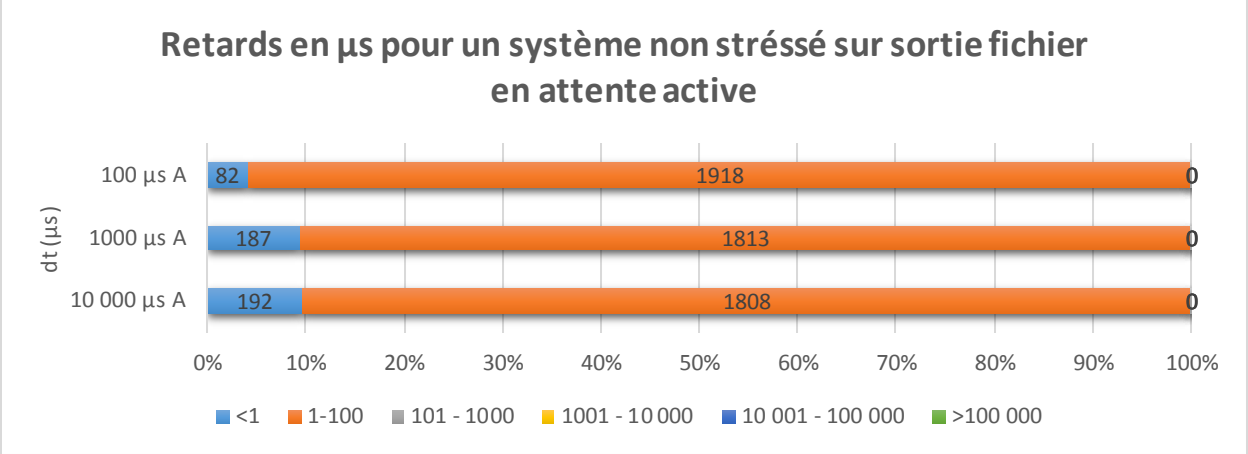
Pendant nos tests, nous avons remarqué que si une tâche annexe venait à s'exécuter au moment de notre expérience, ses résultats pouvaient être faussés. Nous avons donc **testé chaque configuration 4 fois avec quelques secondes d'intervalle.** Il y a donc 2000 tests pour chaque configuration.



Ce premier résultat est déjà très intéressant. Nous remarquons qu'en diminuant le dt, on augmente grandement les retards dans l'attente active. A l'inverse, plus le dt est grand, plus le nombre de retards inférieurs à 1µs augmente. La même expérience avec les attentes passives donne un résultat plus

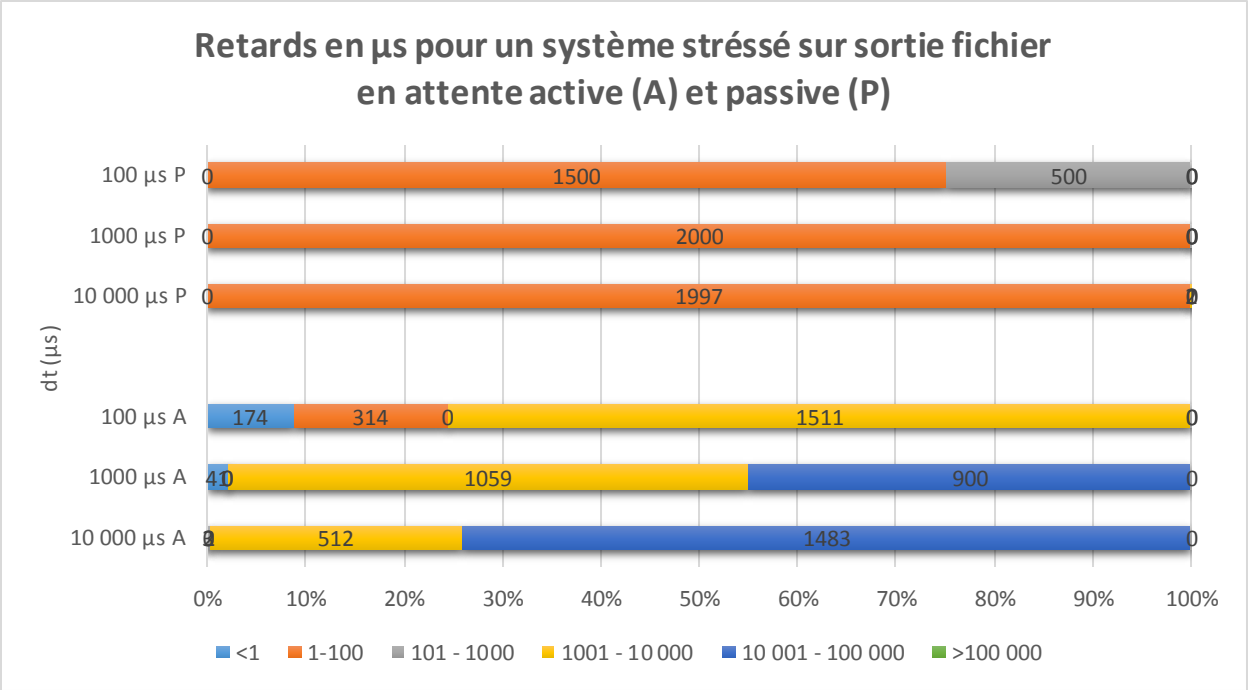
difficilement interprétable. Comme l'échelle de mesure est logarithmique et les temps d'exécution visiblement plus longs, toutes les données sont dans la même tranche de retard.

Nous allons à présent voir l'influence sur les retards de l'écriture dans un fichier pour une attente active. Le fichier est supprimé entre chaque configuration pour prendre en compte son temps de création.



Nous remarquons une légère augmentation des retards et donc une baisse de performance. Mais la différence reste faible, ceci pouvant être lié aux performances de l'ordinateur ou à la faible quantité d'écriture. Le mode passif a été testé et ne présente aucune différence par rapport au graphe page 2.

Pour accroitre la génération de retards, nous allons occuper le CPU. Les résultats suivants reprennent la même configuration que précédemment.



Le mode passif reste extrêmement stable dans ses retards. Bien que ne présentant pas de retard inférieur à 1 microseconde, il réussit, même avec un système très chargé, à les contenir dans la plage des 1-100 µs. Cette stabilité s'explique par l'utilisation de signaux qui sont vus par le système comme étant prioritaires. Ainsi, le processeur délaisse ce qu'il faisait et traite la tâche du programme avant les autres.

Lorsque le système est chargé en mode actif, nous remarquons que moins le programme est stressé, plus le système semble prendre ses aises et néglige le programme. Ce délaissement est lié au coût très élevé d'un changement d'environnement. En conséquence, le processeur favorise des exécutions consécutives venant d'un même programme afin d'accélérer les calculs. A l'inverse, lorsque le programme est stressé, le processeur se force d'exécuter le programme plus régulièrement, diminuant les retards.

3. CONCLUSION

Puisqu'il est possible de générer des retards, il devient évident que Linux n'est pas un système temps-réel. Que ce soit par la verbosité du logiciel, le type d'écriture, la charge de l'OS ou le stress du programme, le processeur alterne entre les tâches afin de donner paradoxalement une impression de temps-réel à l'utilisateur. Si cette configuration convient parfaitement à une utilisation classique d'un ordinateur dans un environnement multitâche, cette impossibilité de prévoir les temps d'exécution peut être problématique dans certains domaines, ou la réactivité et la précision des systèmes sont cruciales.