S. Lavirotte
Polytech Nice Sophia
SI4
2020-2021

TD n° 06
Optimisation d'un Système Embarqué

Le but de ce TD est de mettre en œuvre quelques techniques pour l'optimisation d'un système. Le système que vous devrez optimiser sera basé sur une machine de type Pentium avec 64Mo de RAM.
Vous ferez un compte-rendu des choix et des actions effectuées qui vous ont conduits à un système plus performant.

1 Correction de la configuration du noyau TD02

1.1 Configuration du TD02

Pour pouvoir mesurer les gains que nous pourrons obtenir sur le temps de chargement du noyau, il est nécessaire d'instrumenter celui-ci. Vous partirez de la configuration que nous vous fournissons.

```
cd /work/td02/linux-5.4.91
make mrproper
cp ../config-5.4.91 .config
make oldconfig
```

1.2 Amélioration pour avoir un système qui démarre sans erreur

Afin d'obtenir un noyau qui permet de démarrer sans erreur le système que nous avions lors du TD02, vous allez ajouter les options suivantes dans la configuration :

```
Executable file formats --->
   <*> Kernel support for script starting with #!
File systems --->
   [*] Enable POSIX file locking API
```

Avec ces deux options, l'ensemble des services pourront démarrer sans erreur !

2 Optimisation de la configuration du noyau

Pour ce TD, nous allons donc réutiliser le contenu de l'image disque `sdf-optim` que vous monterez dans `/work/td06`. Cette image contient le noyau `5.4.91` ainsi que l'image d'une machine virtuelle que nous allons tenter d'optimiser.

2.1 Mettre en place les mesures sur le noyau

Pour pouvoir mesurer les gains que nous pourrons obtenir sur le temps de chargement du noyau, il est nécessaire d'instrumenter celui-ci. En partant de la configuration que nous avons préparée à la section 1, nous ajouterez les options permettant d'activer les mesures temporelles dans le noyau. Pour cela, vous vous référerez au cours.

Vous ajouterez aussi

```
Kernel Hacking
   [*] Early printk
   [*] Enable verbose x86 bootup info messages
```

2.2 Evaluation du temps de démarrage d'un noyau

Après cette première configuration, vous configurerez ce noyau `5.4.91` afin d'avoir accès aux mesures temporelles lors du chargement de celui-ci.

Après compilation et génération du noyau, vous démarrerez le système avec votre noyau à l'aide de `qmyqemu.sh`.

Vous ferez alors une première sauvegarde des mesures réalisées du démarrage de ce système. Pour cela, vous utiliserez la commande suivante :

```
dmesg > logs.txt
```

TD n° 06
Optimisation d'un Système Embarqué

Vous utiliserez la commande `scp` depuis votre machine virtuelle de travail pour vous connecter par `ssh` à votre machine virtuelle `qemu` et récupérer le fichier créé via la commande:

```
scp -P 2222 root@localhost:/root/logs.txt .
```

Ce fichier contient les messages de démarrage de votre système avec les informations temporelles.

Vous utiliserez alors le script `bootchart.pl` se trouvant dans les sources du noyau Linux.

```
cat logs.txt | …/scripts/bootgraph.pl > output1.svg
```

Ce graphique vous donnera le temps de démarrage de référence (donc celui à optimiser) pour la configuration du noyau que nous avons. Pour visualisé celui-ci vous le copierez sur votre machine physique via le partage de dossier VirtualBox (`/media/sf_...`).

3 Optimisation de la configuration du noyau

Après la visualisation du résultat obtenu dans `output1.svg`, vous devez constater qu'il est important d'optimiser certaines fonctionnalités très gourmandes en tant d'exécution lors de l'initialisation du noyau. Commencez donc par chercher comment optimiser celles-ci.

3.1 Correction des problèmes les plus évidents

Vous pourrez consulter l'adresse http://elinux.org/Boot_Time pour y trouver des informations sur les actions à potentiellement entreprendre. Mais cette documentation ne correspond pas forcément avec votre version de noyau. Vous confronterez donc les informations obtenues sur ce site avec la documentation de votre version du noyau (`/work/td02/linux…/Documentation`) pour appliquer les bons correctifs.

Appliquez le correctif pour gagner sur le temps de boot du noyau. Vous vérifierez le temps gagné et stockerez votre résultat dans le fichier `output2.svg`.

3.2 Optimisations générales pour la vitesse

Dans un deuxième temps, vous tenterez d'obtenir un gain supplémentaire quant à la vitesse de démarrage de votre noyau. Vous commencerez par simplement ajouter l'option quiet aux paramètres de démarrage du noyau Linux. Vous produirez un fichier `output3.svg` correspondant à l'activation de l'option `quiet`. C'est déjà pas mal de temps gagné juste pour désactiver des messages, non ?

Ensuite, vous activerez les options suivantes pour optimiser votre système en termes de vitesse d'exécution. Vous lirez la documentation des options afin de comprendre leur impact sur le temps de boot du noyau et son impact mémoire.

```
General setup
 - Timers subsystem
  - Old idle dynticks config
  - High Resolution Timer Support
Processor type and feature
 - Processor family (Core 2/newer Xeon) ou (Opteron/Athlon64/Hammer/K8)
 - Timer frequency (1000 HZ)
```

Vous comparerez ce nouveau noyau produit en termes de vitesse de boot toujours à l'aide du script `bootchart.pl`. Vous veillerez bien à ce que les conditions de benchmarking soient identiques d'un test à l'autre afin de ne pas trop influencer sur les mesures effectuées. Vous créerez un fichier `output4.svg` pour cette nouvelle configuration.

Que constatez-vous ?

TD n° 06
Optimisation d'un Système Embarqué

3.3 Optimisations pour la taille du noyau

Il n'est pas tout d'avoir un noyau performant, il faut aussi avoir un noyau incluant toutes les fonctionnalités nécessaires et le plus compact possible. Nous allons travailler ce deuxième critère.

Pour améliorer la taille du noyau (le fichier), vous devez veillez à activer des options dans différentes catégories. Voici les principales. Ceci ne nous empêche pas de lire la documentation d'autres options et d'avoir des gains supplémentaires.

```
General Setup
     - Kernel Compression mode = XZ
     - Kernel log buffer = 12
     - Optimize for size
Processor type and feature
     - High Memory Support (off)
Kernel Hacking:
     - Strip assembleur-generated symbols during link
```

Vous comparerez ainsi la taille du noyau avant et après l'activation de ces options.

4 Optimisation du temps de démarrage des services

A l'aide de l'application `bootchart` présente sur l'image `disk-optim.qcow2`, vous analyserez le temps nécessaire à l'initialisation des services sur votre système.

Pour réaliser ces mesures, vous ajouterez les options suivantes dans la configuration du noyau :

```
Kernel Hacking
 - Collect scheduler debugging info
 - Collect scheduler statistics
```

Puis, pour démarrer avec ce nouveau noyau, vous ajouterez le paramètre suivant : `-append "… init=/lib/systemd/systemd-bootchart"`. Pour visualiser le graphique créé, vous récupérez le fichier `svg` qui se trouve dans `/run/log/bootchar*.svg`.

Que pouvez-vous conclure à la visualisation des résultats obtenus par `bootchart` ? Est-ce que le système utilise déjà un lancement en parallèle des différents services ? Si c'est le cas, que pouvez proposer pour gagner du temps sur le démarrage du système ?

5 Optimisation de la consommation énergétique

Il est aussi possible de configurer votre système pour que celui-ci consomme moins d'énergie. Vous pouvez ainsi autoriser le changement de fréquence du cpu et aussi autoriser les modes de standby ou d'hivernation. Activez ces options dans le noyau.

Vous utiliserez `powertop` sur votre système afin de suivre ses recommandations en termes de configuration de votre noyau et de votre système pour le rendre moins énergivore possible. Pour l'utiliser, il faudra activer :

```
Processor type and Features
 - /dev/cpu/*/msr – Model-specific register support
```

Quel problème rencontrez-vous pour continuer à optimiser votre système ?