UNIVERSITÉ
CÔTE D'AZUR

Introduction aux Systèmes et Logiciels Embarqués

Présentation: Stéphane Lavirotte
Auteurs: ... et al*

(*) Cours réalisé grâce aux documents de :
Bootlin, Stéphane Lavirotte

Mail: Stephane.Lavirotte@univ-cotedazur.fr
Web: http://stephane.lavirotte.com/
Université Côte d'Azur

UNIVERSITÉ CÔTE D'AZUR

Pourquoi donc Optimiser ?

- ✓ Une problématique pas si nouvelle…
- ✓ Steve Jobs s'adressant à Larry Kenyon (ingénieur « driver disk » et « file system »)
  - « *The Macintosh boots too slowly. You've got to make it faster!* »
  - …
  - « *Well, let's say you can save 10 seconds off of the boot time. Multiply that by five million users and thats 50 million seconds, every single day. Over a year, that's probably dozens of lifetimes. So if you make it boot ten seconds faster, you've saved a dozen lives. That's really worth it, don't you think?* »
- ✓ Au final, cela fait gagner beaucoup de temps à tous !

« Bâtisseurs d'empire par accident », Robert X. Cringely
« Revolution in the Valley », Andy Hertzfeld

UNIVERSITÉ CÔTE D'AZUR

Optimiser Quoi ?

- ✓ Encore plus important pour un système embarqué
  - Démarrage:
    - Quelques minutes sur un PC, « quelques » secondes sur un SE
  - Des plateformes moins performantes
  - Des ressources limitées !
- ✓ Les pistes pour améliorer les performances d'un système
  - Augmenter la vitesse de production
    - Pour produire plus vite le système souhaité
  - Augmenter la vitesse d'exécution du système sur la cible
    - Pour avoir un noyau plus efficace en temps de chargement
    - Pour des services qui démarrent plus vite et des applications plus rapides
  - Réduire la taille: emprunte disque et mémoire
  - Réduire la consommation énergétique
- ✓ Conclusion

UNIVERSITÉ
CÔTE D'AZUR

Augmenter la Vitesse de Compilation

Pour obtenir un système plus rapidement

UNIVERSITÉ CÔTE D'AZUR

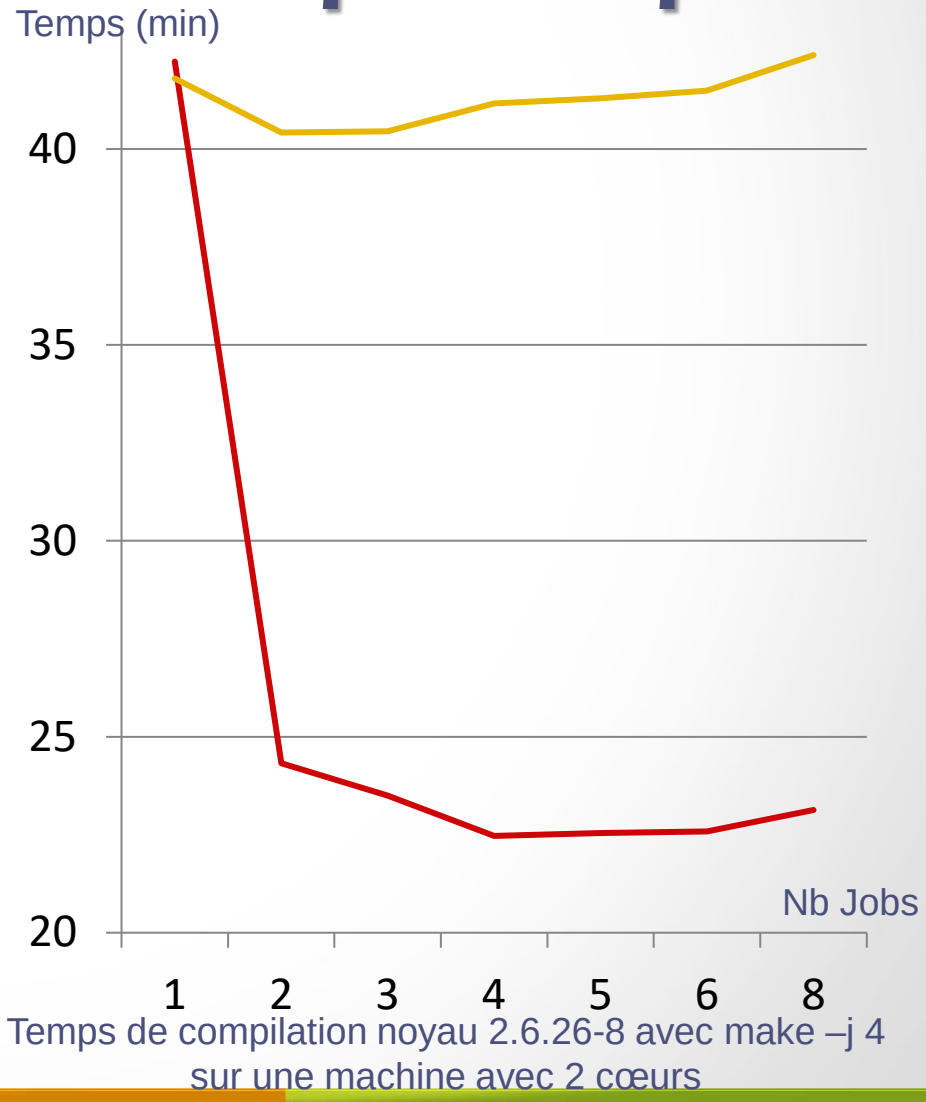
Tirer le Meilleur parti de sa Machine de production

- ✓ Ne pas charger la machine avec d'autres processus
- ✓ Si en machine virtuelle:
  - Tirer le meilleur parti de sa machine physique...
    - Etre si possible en virtualisation et pas en émulation !

UNIVERSITÉ CÔTE D'AZUR

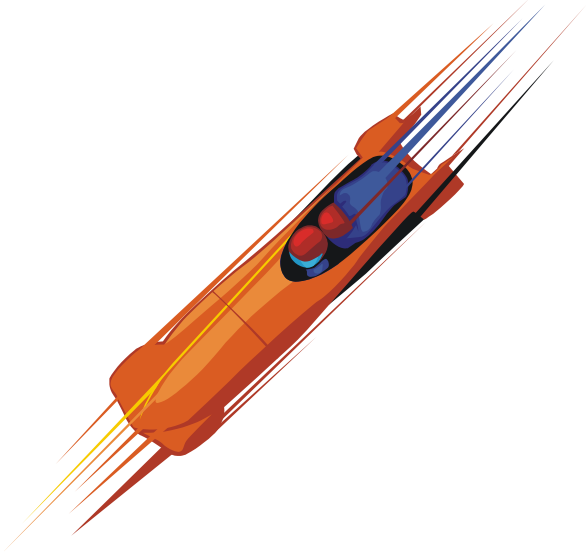
Plusieurs Options pour une Compilation plus rapide

- ✓ -pipe
  - Utilise des tubes à la place de fichiers temporaires.
  - Pas d'effet sur le code produit
- ✓ –j
  - Spécifie le nombre de jobs simultanés
  - Gains
  - Attention: plusieurs traces, donc erreurs pas en dernier



Temps de compilation noyau 2.6.26-8 avec make –j 4 sur une machine avec 2 cœurs

UNIVERSITÉ
CÔTE D'AZUR



Augmenter la Vitesse de Démarrage

Pour démarrer le noyau plus rapidement

UNIVERSITÉ CÔTE D'AZUR

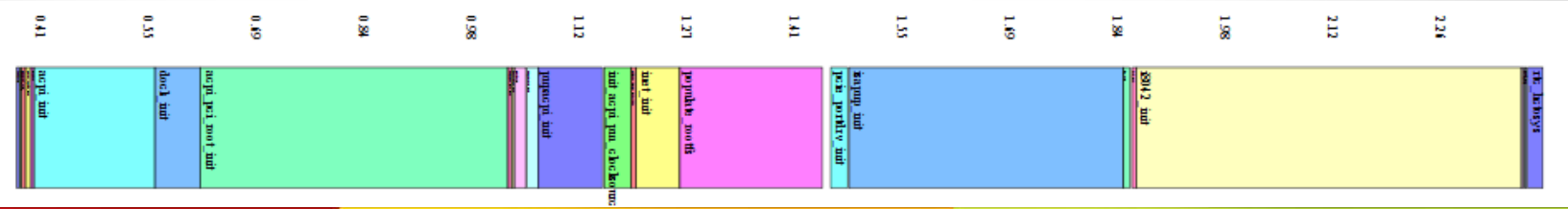
Faire des Mesures sur le Noyau

- ✓ `CONFIG_PRINTK_TIME` (section Kernel Hacking)
  - Introduit depuis 2.6.11
  - Ajoute des informations temporelles dans les messages
  - Se configure dans la section Kernel Hacking
  - Simple et Robuste
- ✓ Limitations:
  - Peut causer des problèmes sur certaines plateformes
  - Pas de précision sur certaines plateformes
    - Utilise la routine sched_clock()
    - A une résolution de 1 jiffy (pouvant être de 10ms)
    - Très bonne résolution à 1ms (nécessite une fréquence de 1MHz)
- ✓ Programmes utilitaires dans les scripts du noyau

UNIVERSITÉ CÔTE D'AZUR

Boot Tracer

- ✓ `CONFIG_FUNCTION_TRACER` *(section Kernel Hacking)*
  - *Introduit depuis 2.6.28*
  - *Permet de mesurer le temps des initcalls*
- ✓ *Pour activer cette fonctionnalité*
  - *Le noyau doit avoir été compilé avec ces options*
  - *Démarrer le noyau avec les options suivantes*
    - `initcall_debug` **et** `printk.time=1`
- ✓ *Pour visualiser ces informations*
  - *Génération d'un fichier SVG*
  - `dmesg | perl scripts/bootgraph.pl > output.svg`



UNIVERSITÉ CÔTE D'AZUR

Désactiver les Messages sur la Console

- ✓ Les messages de boot du noyau sur la console
  - L'affichage prend du temps
  - Le scrolling du frame buffer a un coût !
  - Ces messages ne servent à rien dans un système en production
- ✓ Désactivation de ces messages
  - Maintenant le cas dans la plupart des distributions standards
  - Il suffit de rajouter l'option `quiet` au boot
- ✓ Benchmark:
  - Peut réduire le temps de démarrage de 30 à 50%
  - Soit environ 200ms

```
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009fc00 (usable)
 BIOS-e820: 000000000009fc00 - 00000000000a0000 (reserved)
 BIOS-e820: 00000000000ce000 - 00000000000d2000 (reserved)
 BIOS-e820: 00000000000f0000 - 0000000000100000 (reserved)
 BIOS-e820: 0000000000100000 - 000000003fff0000 (usable)
 BIOS-e820: 000000003fff0000 - 000000003fff8000 (ACPI data)
 BIOS-e820: 000000003fff8000 - 0000000040000000 (ACPI NVS)
 BIOS-e820: 00000000fff00000 - 0000000100000000 (reserved)
127MB HIGHMEM available.
896MB LOWMEM available.
On node 0 totalpages: 262128
zone(0): 4096 pages.
zone(1): 225280 pages.
zone(2): 32752 pages.
Kernel command line: BOOT_IMAGE=linux ro root=302 BOOT_FILE=/boot/vmlinuz-2.4.18-17
.7.x 1
Initializing CPU#0
Detected 1992.572 MHz processor.
Speakup v-1.00 CVS: Tue Jun 11 14:22:53 EDT 2002 : initialized
Console: colour VGA+ 80x25
Calibrating delay loop... 3961.24 BogoMIPS
Memory: 1027712k/1048512k available (1153k kernel code, 17216k reserved, 975k data,
 160k init, 131008k highmem)
```

Jiffy

- ✓ Jiffy
  - Exprime une période de temps courte (voire très courte)
- ✓ Utilisé dans différents domaines
  - En électronique:
    - période de temps entre l'alternance du courant (1/60 ou 1/50 de seconde) (20ms)
  - En Informatique:
    - Temps entre deux interruptions timer
    - Dépendant de l'architecture et de la configuration noyau (constante HZ)
      - Depuis Linux 2.6.20: peut aller de 1 à 10ms: HZ= 100, 250 (par défaut), 300 or 1000
      - Donc par défaut sur x86: 1/250 de seconde donc 4ms (10ms sur ARM)
  - En Physique
    - Temps mis par la lumière pour parcours une distance donnée
    - Pour parcourir un femi (10^{-15}m) : 3×10^{-24} secondes

UNIVERSITÉ CÔTE D'AZUR

Pré-réglage LPJ: loops_per_jiffy

- ✓ A chaque démarrage du noyau calcul du LPJ
  - Exécute la fonction `calibrate_delay` pour le calibrage de la boucle de délai (utilisé pour la fonction `udelay`)
  - Cette valeur est `loops_per_jiffy` (LPJ)
  - Prends environ 250 ms sur la plupart des hardware embarqués (25 jiffies, avec pour ARM 1 jiffy = 10ms)
- ✓ Pré-réglage de la valeur LPJ pour éviter son calcul
  - Dans les messages de boot du noyau retrouver la valeur
    - `Calibrating delay loop... 187.59 BogoMIPS (lpj=937984)`
    - Si ce message n'apparaît pas dans la console démarrer le noyau avec `loglevel=8`
  - Pour les prochains démarrages ajouter lpj=<*value*>
    - Vous pourrez noter le nouveau message:
      - `Calibrating delay loop (skipped)... 187.59 BogoMIPS preset`

UNIVERSITÉ CÔTE D'AZUR

Copie DMA ou XIP

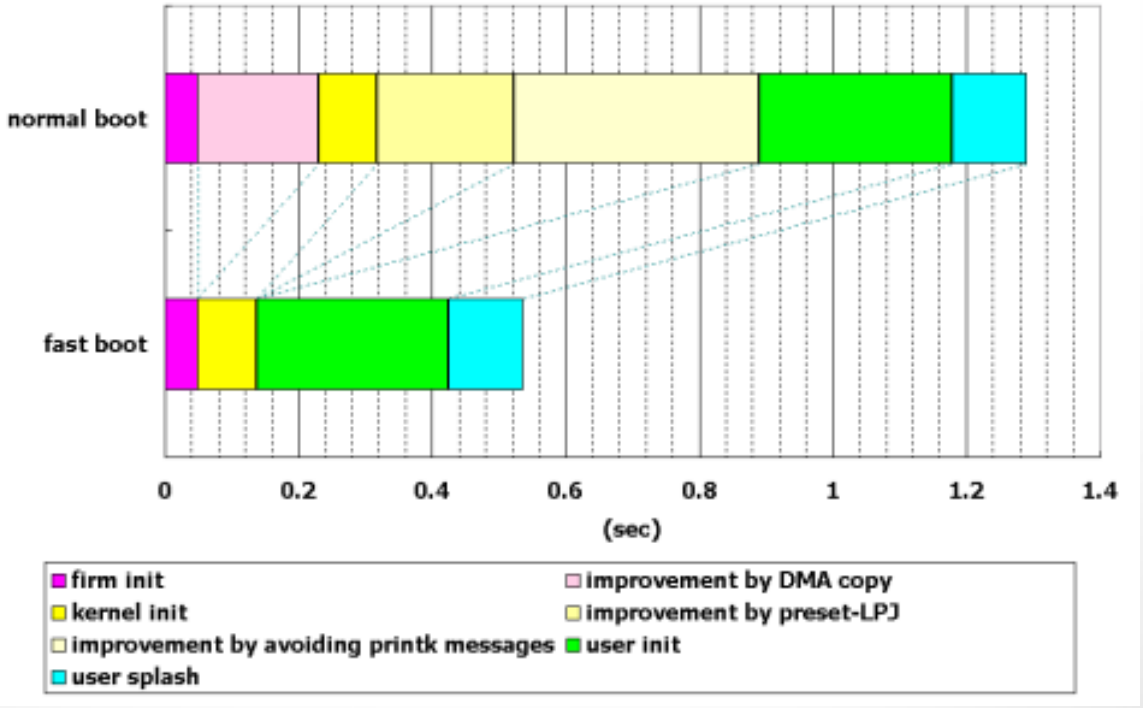
- ✓ Pour que le noyau s'exécute:
  - Lecture du noyau depuis la flash ou ROM
  - Décompression du noyau
  - Écriture du noyau en RAM
- ✓ 2 types d'optimisations possibles:
  - Copie DMA permet d'accélérer grandement le temps de chargement
  - « Execute in Place » (XIP)
    - Inclus depuis le noyau 2.6.10
    - Mise en place dépendante de la plate-forme cible
      - http://elinux.org/Kernel_XIP_Instructions_For_OMAP

UNIVERSITÉ CÔTE D'AZUR

Conclusion des Optimisations pour le Démarrage du Noyau

✓ Gains cumulés par ces différentes optimisations:

	Firm init	Kernel init	User init	User splash	Total
Normal boot	229	660	290	112	1 291
Fast boot	49	88	287	113	537



UNIVERSITÉ CÔTE D'AZUR

D'Autres Idées d'Optimisations

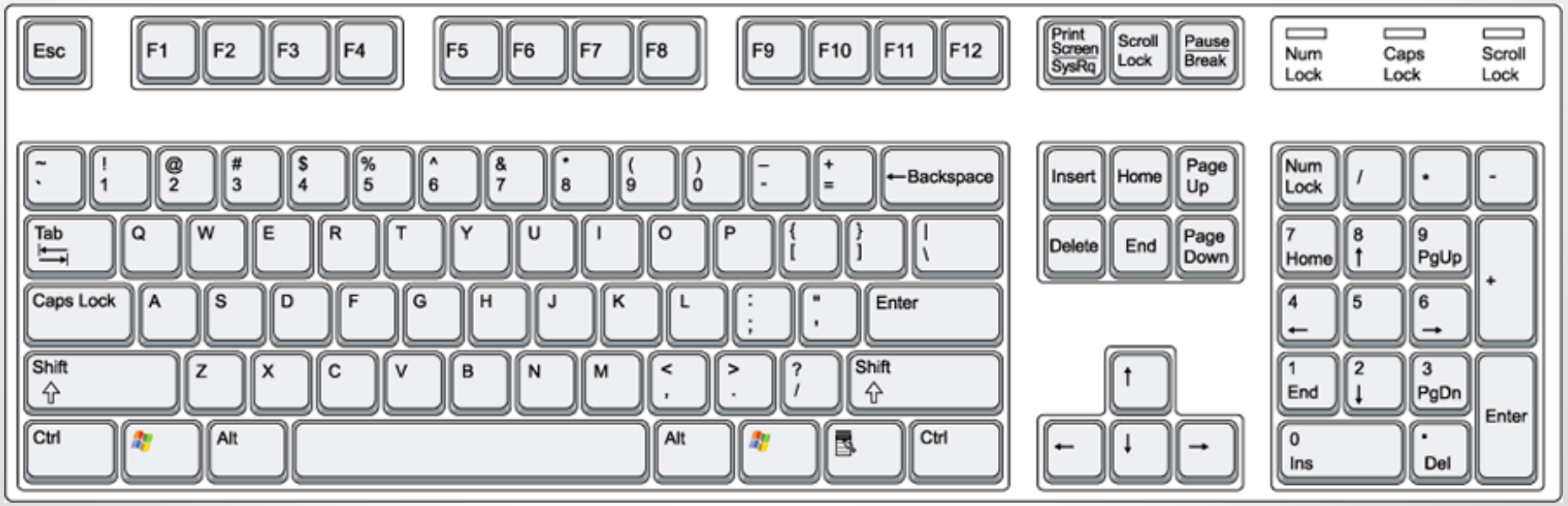
- ✓ Diverses autres optimisations sont possibles:
  - Mettre ce qui n'est pas nécessaire au boot en modules
    - Permet d'avoir un noyau plus petit, donc plus rapide à copier en RAM
    - Permet d'éviter l'initialisation des modules au boot
  - Spécifier l'allocation de mémoire au boot
  - Supprimer tout ce que le système n'utilise pas
    - Si pas besoin de sysfs ou procfs, gain de 20ms
  - Réduire le temps d'investigation pour les opérations IDE
  - Supprimer le délai pour synchroniser l'horloge système avec le RTC au démarrage

- ✓ Allez consulter le site elinux.org section Boot Time

UNIVERSITÉ CÔTE D'AZUR

Tester les Options de Boot du Noyau Linux

- ✓ Modifier les options de boot du noyau
  - Qemu: ajouter `-append "..."` sur la ligne de commande
  - Grub: tester avant d'intégrer les options dans `grub.cfg`
    - Problème:
      - Forcément un clavier en anglais
      - Layout d'un clavier qwerty pour vous aider !



Augmenter la Vitesse du Système

Pour démarrer le système plus rapidement

UNIVERSITÉ CÔTE D'AZUR

Optimiser les Scripts

- ✓ Comment optimiser les/ses scripts pour BusyBox ?
  - Diminuer le nombre de fork/exec
- ✓ Règles de mise en œuvre:
  - Ne pas mettre de code inutile (!)
  - Remplacer les commandes et utilitaires externes par ceux de Busybox (builtins)
  - Ne pas utiliser de commande en pipe si possible
  - Réduire le nombre de commandes en pipe
  - Minimiser l'utilisation des commandes en « backquote »: ` `
- ✓ Voir les exemples :
  - http://elinux.org/Optimize_RC_Scripts

UNIVERSITÉ CÔTE D'AZUR

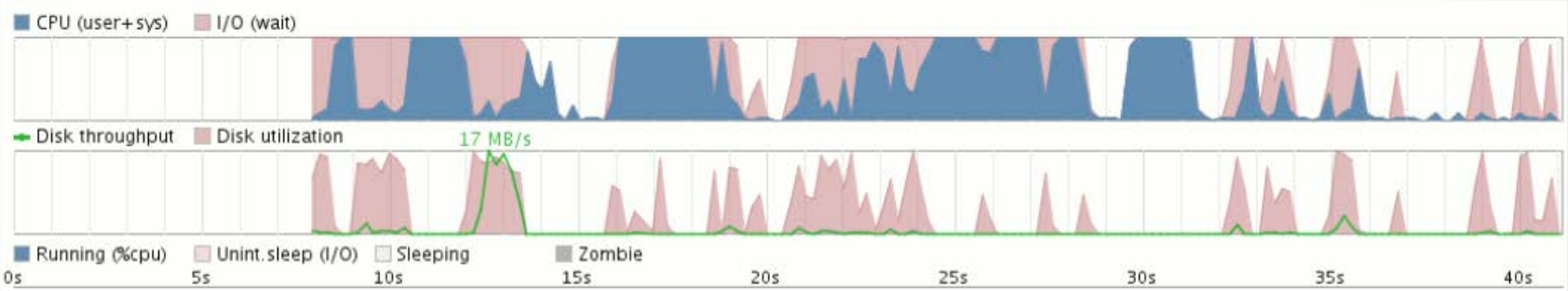
Démarrage des Services

- ✓ Initialisation SysV:
  - Démarrage des services séquentiellement
  - Attend la fin du script courant pour démarrer le suivant
  - Des dépendances existent dans le démarrage des scripts
    - Démarrage du service réseau avant de démarrer un serveur...
  - Mais de nombreuses tâches pourraient être exécutées en parallèle
- ✓ Donc gain de temps important possible !
  - Parallélisation du lancement des scripts: systemd
- ✓ Analyse du démarrage: bootchart
  - Remplace le processus de lancement: `init=/sbin/bootchard`
  - Après démarrage, exécuter: `bootchart`
    - Génère des graphiques sur les temps de démarrage des scripts

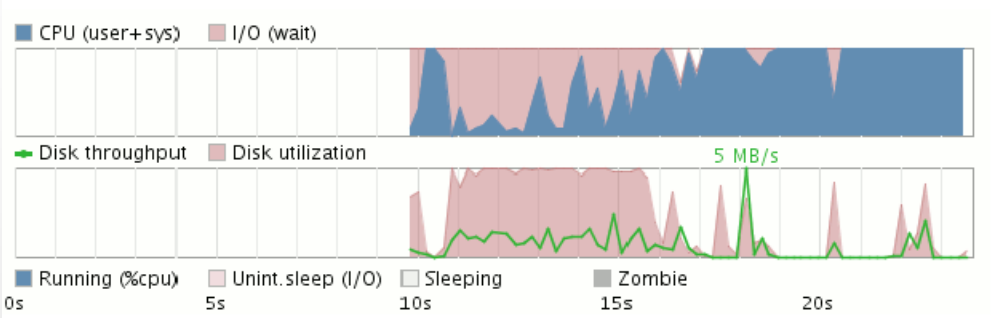
UNIVERSITÉ CÔTE D'AZUR

Exemple de Gain

- ✓ Démarrage sans et avec parallélisation
  - Démarrage des services sans parallélisation



  - Démarrage des services avec parallélisation



Graphiques réalisés avec [Bootchart]

- ✓ Près de 2 fois plus rapide !

UNIVERSITÉ CÔTE D'AZUR

Système en Hibernation

- ✓ Sauvegarde de l'état du système
  - Nécessite le support noyau et matériel
  - En RAM: continue à consommer de l'énergie
  - Sur disque: nécessite de l'espace de stockage
- ✓ Avantages
  - Evite le chargement étape par étape du début à la fin
  - Restaure l'état du système tel qu'il était à l'extinction
- ✓ Inconvénient:
  - L'écriture sur Flash est plus lente que la lecture
    - Nécessite un peu de temps à l'extinction
- ✓ Concrètement utilisé:
  - Appareils Samsung: DTV, Téléphone Mobiles, Appareils photo-numériques photographique Samsung (800ms)

UNIVERSITÉ
CÔTE D'AZUR

Augmenter la Vitesse des Applications

Pour des application plus rapides

UNIVERSITÉ CÔTE D'AZUR

De Nombreuses Options de Compilation: make -...

- ✓ -O
  - 0: désactive complètement les optimisations
  - 1: niveau classique d'optimisation
  - 2: niveau recommandé: augmenter les perfs du code sans faire de compromis sur la taille
  - 3: le plus élevé et le plus risqué. Pas recommandé avec gcc 4.x
  - S: optimise la taille du code. Active toutes les options de 2 qui n'influent pas la taille du code. Peu causer des problèmes. Pas recommandé
- ✓ Trop d'options tue l'option:
  - funroll-loops, funroll-all-loops, ffast-math, fforce-mem, fforce-addr: peut avoir l'effet contraire souhaité
  - fomit-frame-pointer: réduit la taille du code: déjà activé par –O
- ✓ Faites confiance à l'auto-configuration ou au Makefile

UNIVERSITÉ
CÔTE D'AZUR

Exécutables Statiques

- ✓ Permet de gagner du temps
  - Pas d'édition de liens dynamiques
- ✓ Valable pour des systèmes avec peu de fichiers
  - Systèmes avec 1 à 2 Mo de fichiers et peu d'exécutables

UNIVERSITÉ CÔTE D'AZUR

Exécutables avec Librairies Partagées

- ✓ Pre-Linking
  - Quand une application est lancée, nécessiter de résoudre ces symboles externes
    - Recherche dans la table des symboles des librairies partagées
    - Et modifie le programme binaire pour faire référence à l'offset correct dans la librairie
  - Economiser le temps de ce travail au runtime
- ✓ Avantages:
  - Permet d'économiser beaucoup de temps sur les grosses applications
  - Permet aussi de sauvegarder de la mémoire (réallocation)
- ✓ Inconvénient:
  - Doit être exécuté à chaque fois qu'une librairie partagée change (peu souvent le cas sur un système embarqué)

Réduire la Taille

Emprunte mémoire et Utilisation disque

UNIVERSITÉ CÔTE D'AZUR

Réduire la Taille du Noyau

- ✓ Linux Tiny (maintenu par Bootlin):
  - Réduire l'emprunte mémoire et la taille sur le disque
  - Un ensemble de patchs sur les sources noyau
  - De nombreuses fonctionnalités ont déjà été incluses
- ✓ Optimisations
  - Suppression des messages du noyau
  - Suppression de certaines fonctionnalités non utiles pour une architecture donnée
  - Réduction de la taille de certaines structures (peu influencer les performances)
- ✓ Utilisation
  - Option à activer à la configuration du noyau
    - `CONFIG_EMBEDDED`

UNIVERSITÉ CÔTE D'AZUR

Exécutables avec Librairies Partagées

- ✓ Utilisation des librairies partagées
  - Avantages:
    - Exécutables plus petits (pas dans chaque exécutable
    - Economise de l'espace en RAM par partage des librairies
    - La librairie est copiée une fois
  - Inconvénients:
    - Linkage dynamique au runtime donc applications plus lentes à charger

UNIVERSITÉ CÔTE D'AZUR

Utilisation d'une Librairie C plus Compacte

- ✓ glibc
  - La plus standard
  - Mais taille importe pour supporter l'ensemble des fonctionnalités
  - Environ 1.7Mo sur ARM
- ✓ uClibc
  - Quelques simplifications par rapport aux standards
  - Disponible sur de plus en plus de systèmes embarqués
  - Environ 400Ko sur ARM

- ✓ Particulièrement intéressant en cas d'exécutable compilés statiquement

UNIVERSITÉ CÔTE D'AZUR

Stripper les Exécutables

- ✓ Les programmes compilés inclus des informations
  - Utile pour le développement mais pas pour l'utilisation finale
- ✓ Pour supprimer ces informations
  - Utilisation de la commande `strip`
    - Sur les exécutables ET les librairies
  - Utilisation de la commande `findstrip`
    - Permet de localiser tout ce qui doit être strippé
- ✓ Obtenir des informations sur un exécutable
  - Commande `file`
    - `hello: ELF 32bit LSB executable, ARM, version 1 (SYSV), for GNU/Linux 2.6.26, dynamically linked (uses shared libs), stripped`

UNIVERSITÉ CÔTE D'AZUR

Super Strip

- ✓ `sstrip`
  - Aller encore plus loin que la commande `strip`
  - Permet de gagner encore quelques octets non utilisés par Linux pour démarrer les exécutables
  - Peut aussi être utilisé sur les librairies
    - Limitation: ne peut plus utiliser la librairie pour le développement
  - Très performant en particulier pour les petits exécutables

UNIVERSITÉ
CÔTE D'AZUR

Systèmes de Fichiers

- ✓ Utilisation de systèmes de fichiers compressés pour gagner de l'espace de stockage
- ✓ Pour le système de fichiers en lecture seule
  - Utilisation de SquashFS
- ✓ Pour les systèmes de fichiers en lecture/écriture
  - Utilisation d'UBIFS (ou JFFS2) pour les petites partitions

UNIVERSITÉ
CÔTE D'AZUR

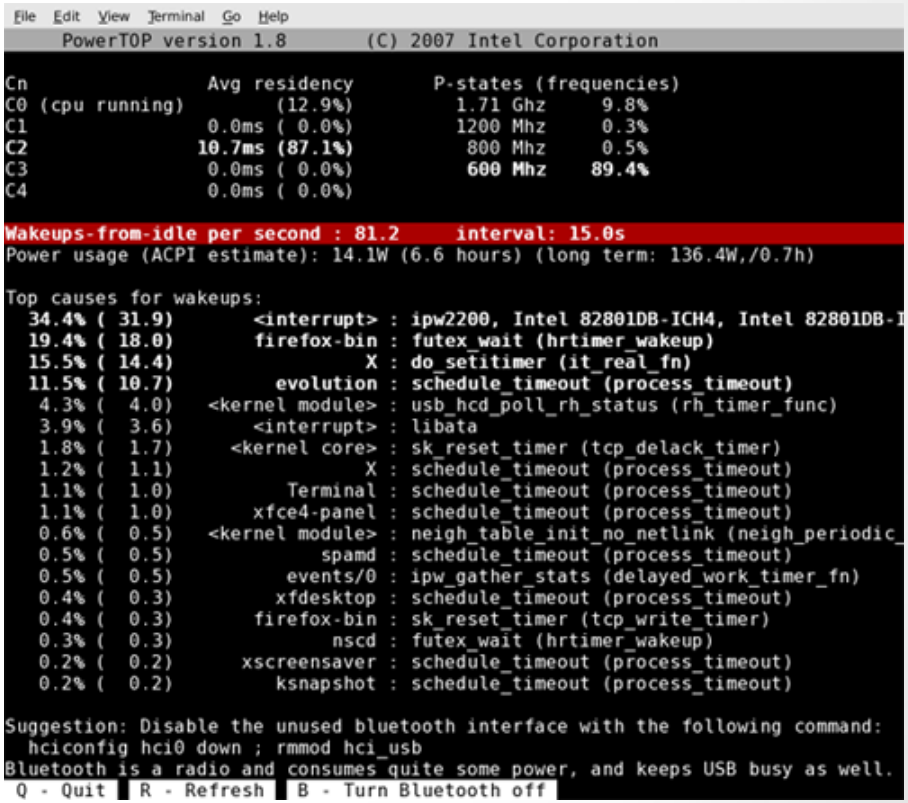
Economie d'Energies

The Green Power !

UNIVERSITÉ CÔTE D'AZUR

PowerTOP

- ✓ PowerTop
  - Un utilitaire montrant les 10 sources de consommation
  - Noyau > 2.6.11 et x86
  - Mesure le nombre de `wake_up` et compte le temps passé en mode `low_power`
  - Fournit des conseils



UNIVERSITÉ CÔTE D'AZUR

cpufreq

- ✓ CPU_FREQ
  - Activer dans les options « Power Management » du noyau
  - Permet de changer la fréquence du CPU « au vol »
  - Architectures supportées: x86, x86_64, ARM, Blackfin, PowerPC, Sparc64
- ✓ Avantages:
  - Configurable depuis l'espace utilisateur (via /sys)
  - De nombreux utilitaires pour la gestion:
    - cpufreqd, cpufrequtils, cpuspeed, ...
  - Economise vraiment les batteries sur un portable

Conclusion

Le mot de la fin

UNIVERSITÉ CÔTE D'AZUR

Conclusion

- ✓ De nombreuses optimisations possibles
- ✓ De vrais gains à réaliser en terme de:
  - Temps
    - Chargement du système
    - Exécution du système
  - Espace de stockage
  - Consommation électrique
- ✓ Mais
  - Des choix cornéliens certaines fois
    - Compromis vitesse / espace mémoire ou de stockage