Systeme de fichiers pour NAND Flash

Corentin Chary p0404514

26 juillet 2007

Table des matières

Chapitre 1

Remerciements

- À la balle jaune pour avoir égaillé nos journées
- À la crêperie, au kebab et a la boulangerie.
- À Mr Guillaume BERNARDINI pour sa rigueur militaire
- À Mr Gregory “Du Gourdin” LEMERCIER pour ses attributs
- À Mr Pierre GAL pour avoir mis la première balle sur le toit
- À Mr Vincent BELLOIR pour être parti
- À Mr Loïc “Loutre” Tourlonias pour ses musiques exotiques
- À Mr Jean-Jacques HIBLOT pour m’avoir fait refaire NandFS trois fois.
- À Mr Pierre “Casse Bonbon” pour ses remarques perspicaces.
- À Mr Alban Marie LIMONET pour son genou bionique.
- À Mr Vasken GUZEL pour nous avoir retransmis le savoir de Patrick BRUEL.
- À Mr Dominique MOREL et Gwendoline AUVIGNE pour nous avoir fournis en bonbons
- À tout le département logiciel pour sa bonne humeur.
- Enfin, je n’oublie pas non plus tout le reste du personnel de l’entreprise pour son amabilité et son aide à mon égard.

Chapitre 2

Stage

- Maître de stage Entreprise : HIBLOT Jean-Jacques
- Tuteur UCBL : BOUNEKKAR Ahmed

2.1 Problématique

Les systèmes embarqués utilisent de plus en plus la mémoire Flash NAND pour stoker les données car cette mémoire a de multiples avantages. Mais cette mémoire a aussi des limitations, et nécessite donc pour utilisation viable, des systèmes de fichiers spécifiquement créés pour prendre en compte ces limitations.

Actuellement, sous Windows, le système de fichiersutilisé est FAT. Ce système de fichierscréé à la base pour des disques dur pose des problèmes de corruption lors de coupures de courant, et est peu performant. Il existe d'autres solution sous Windows, mais elles restent très onéreuses.

2.2 Sujet

Système de fichier pour mémoire FlashNAND pour stockage de données en milieu industriel environnement ARM9+WindowsCE

2.3 Conditions

J'ai eu la chance de pouvoir commencer ce projet. J'ai travaillé en autonomie et seul sur ce projet, mais bien sûr encadré par Mr Jean-Jacques HIBLOT, et aidé par les autres personnes du département logiciel.

Chapitre 3

Environnement : Adeneo

3.1 L'Entreprise

ADENEO est une société d'ingénierie, de conception et d'industrialisation de produits électroniques dont le siège social est basé à Ecully, dans la région lyonnaise.

C'est une société anonyme au capital de 373 920 détenu majoritairement par des dirigeants actionnaires. Depuis sa création en 1990, le chiffre d'affaires de l'entreprise est passé de 610 Mille euros à 50 Millions d'euros aujourd'hui, et le nombre de salariés est passé de 7 à 500 personnes.

Depuis le 1Re janvier 2005, les sociétés du groupe, ADETEL pour la partie électronique, ADESET pour les logiciels embarqués et ADETEST pour les essais aggravés, ont fusionné pour devenir ADENEO.

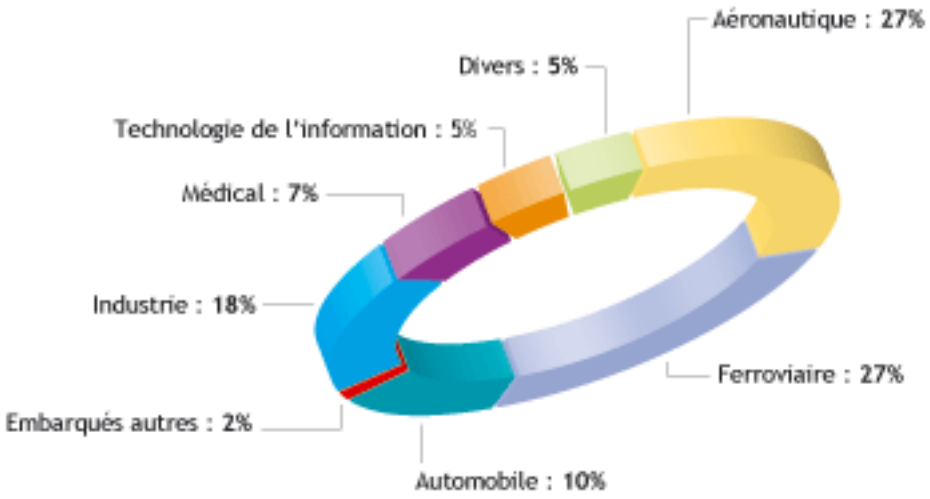
La société ADENEO a atteint le niveau de "Gold-level Member" c'est-à-dire le plus haut niveau Microsoft sur Windows CE. Seules dix entreprises en Europe dont deux en France ont reçu cette distinction de la part de Microsoft.

3.1.1 Implantation géographique

- Ecully : Siège Social
  - 150 personnes - 2000 m^2
  - 2 Chemin du Ruisseau - 69136 Ecully Cedex
  - 04 72 18 08 40 - Fax : 04 72 18 08 41
- Toulouse : Agence
  - 20 personnes - 200 m^2
  - 239 B Route de Saint Simon - 31100 Toulouse
  - 05 34 60 31 58 - Fax : 05 34 60 29 97
- Paris : Agence
  - 10 personnes - 170 m^2
  - 2 Rue Paul Cézanne - 93360 Neuilly-Plaisance
  - 01 43 09 73 35 - Fax : 01 43 09 71 94

– Grenoble : Centre de validation de la Conception
  – 20 personnes - 500 m^2

3.1.2 Domaines D'Activités



Applications Aéronautiques

- Avionique civile : A380, 787.
- Avionique militaire : A400M, Mirage 2000, Rafale, système d'armes.
- Équipements héliportés.
- Logiciels DO178B jusqu'au Level A, électronique DO254 niveau C, FPGA DO254 niveau C.
- Sous ensemble d'interconnexion de calculateurs embarqués (¿3 000 points, vérification automatique de la CAO)

Applications Ferroviaires

- Équipements pour matériel roulant du Tramway au TGV.
- Équipements de signalisation.

Applications Automobiles

- Équipement de localisation GPS pour véhicule.

Applications Industrielles

- Système de télé-relevé et de maintenance sur les compteurs d'énergie sous Windows CE.

– PDA durci avec fonction GPS sous Windows CE.
– Calculateur d'E/S SIL2.

Applications Médicales

– Système de radiographie numérique dynamique.
– Système de radiographie dentaire.
– Logiciel de système d'assistance respiratoire
– Interface pour visualisation sur PC d'un échographe oculaire.

Applications Aéronautiques

– Convertisseurs statiques de 1,5 à 15KVA, à partir du 115V fréquence variable ou du 28 V CC pour ventilation / climatisation et inverseurs de poussée.
– Alimentations 100 W à 4 kW pour actionneurs et commande électrovannes.
– Chargeurs de batterie 48V - 1500VA pour moyen d'essai en vol (alimentation secourue).

Applications Ferroviaires

– Convertisseurs statiques de 5 à 100KVA, de 750 à 3000Vdc pour ventilation et compresseur / climatisation.
– Chargeurs batteries embarqués : 1000V - 7 KVA - Batteries au plomb, NiMH, SuperCap pour énergie traction.
– Alimentation BT de système contrôle commande.

Applications Marines

– Contrôle commande d'un système d'armes navals.
– Convertisseurs statique DC/DC 115V 400Hz pour alimentation BT.

3.1.3 Les différentes prestations

Possédant des compétences variées, ADENEO propose différents types de prestations : l'assistance technique (AT), le forfait et la prestation d'étude et fourniture.

Prestations en Assistance Technique

Le but de l'assistance technique est de faire travailler un ingénieur ou un technicien ADENEO directement chez un client pour lui apporter un support technique dans son développement. Le client assure la logistique durant la mission, qui peut durer de quelques semaines à plusieurs années.

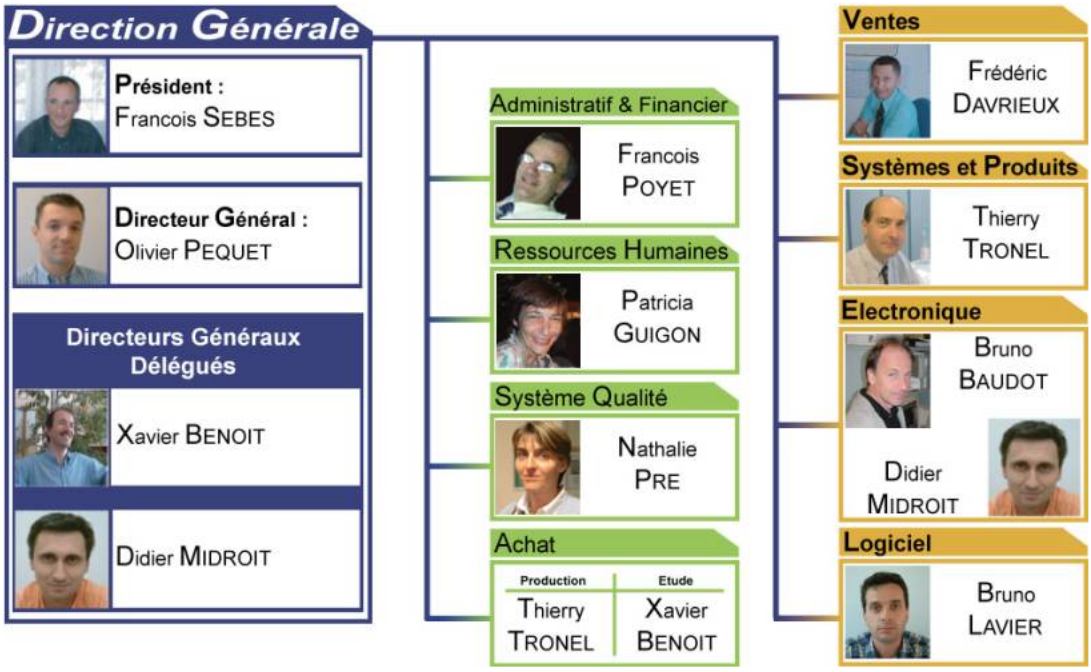
Prestations au Forfait

Contrairement aux prestations en AT, celles-ci se déroulent dans les locaux d'ADENEO. Une prestation est définie avec le client par le biais d'un cahier des charges. ADENEO propose une solution technique chiffrée en termes de coût et de délais, c'est le devis.

Prestations d'Étude & Fourniture

ADENEO s'engage à livrer les produits issus de l'étude (pré-série ou série). Le coût de l'étude est partiellement, voir totalement, pris en charge par ADENEO. Elle est financée par la fourniture des produits. Il peut également s'agir de projet R&D (Recherche et Développement), mon sujet de stage.

3.1.4 Organisation



3.1.5 Départements

La société ADENEO est divisée en trois départements, rattachés à la direction générale, qui sont Service & Produit, Logiciel et Électronique.

Le département Service & Produit

- Ce département s'occupe de l'industrialisation, de la gestion de production, mais également :
- Des études en électronique de puissance,
- Des études thermiques,
- De l'activité banc de test et simulateurs,
- Des essais aggravés et de vieillissements.

Le département Logiciel

- Les compétences du département logiciel sont :
- La réalisation de logiciel embarqué, avec ou sans OS,
- La formation sur Windows et Linux Embedded,
- La programmation de FPGA,
- La conception d'IHM.

Le département Électronique

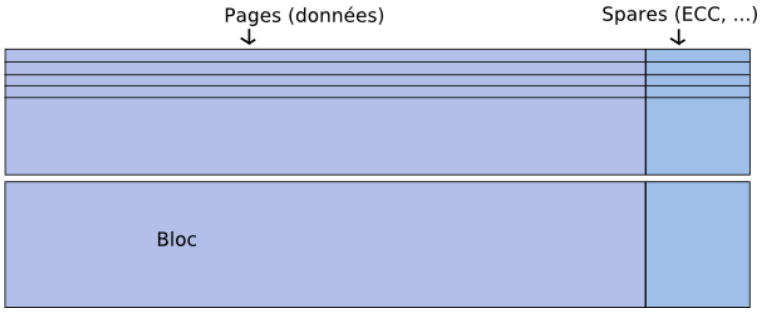
- Le département électronique, réalise des développements de systèmes analogiques, numériques et microélectroniques.

Chapitre 4

Travail Effectué

4.1 Documentation : NAND

La première chose que j'ai du faire a été de me documenter sur le fonctionnement des NAND Flash. En effet ce type de mémoire a un fonctionnement bien particulier. Un NAND Flash peut être vue comme une suite de *blocs d'effacement*, chaque bloc contenant un certain nombre de *pages d'écriture*, chacune de ces pages associée à une zone de spare.



Une page ne peut être écrite qu'une seule fois, pour être réécrite, son bloc doit être effacé (donc toutes les pages contenues dans ce bloc). Ainsi, il est impossible de modifier une page sans effacer tout un bloc. La zone de spare permet de stoker des informations sur l'état du bloc et des codes de correction d'erreur (ECC). En effet, au bout d'un certain nombre d'effacement (100 000) les blocs se détériorent, et il est nécessaire de les marquer, et de corriger les erreurs si il y'en a.

Il existe deux types de NAND distincts :

- Small Pages : pages de 512 octets, 16 octets de spare, 32 pages par bloc.
- Large Page : pages de 2048 octets, 64 octets de spare, 64 pages par bloc.

4.2 Échauffement : NAND et WinCE

Après m'être documenté sur la NAND, j'ai été chargé de reprendre le driver de NAND existant pour Windows CE. Ce driver, fonctionnant sur les cartes ATMEL AT91SAM960/AT91SAM962/AT91SAM963 possèdait un fichier spécifique pour chaque modèle de NAND, et ce pour chaque carte. Le but de mon travail était de refactoriser (regrouper les parties communes) le code. Cela m'a permit de mettre en pratique ce que j'avais lu sur les NAND, et de découvrir Windows CE et les outils associés (Visual Studio 2005). Afin de comprendre le fonctionnement exact de chaque type de NAND et pour chaque opération (lecture, écriture, effacement), j'ai eu l'occasion de lire les Datasheet fournies par les constructeurs.

4.3 Documentation : NAND FSs

Une fois le fonctionnement d'une NAND Flash bien assimilée, j'ai commencé à chercher des informations sur les systèmes de fichiers existants.

Le résultat détaillé de cette recherche est disponible dans l'*annexe 5*

Pour résumer sous Windows, il existe peux de solution : en général c'est FAT qui est utilisé, avec une couche d'abstraction, les autres FS sont payants. Sous linux, il existe trois solutions :

- JFFS2, créé à la base pour les NOR, peut performant, mais gère la compression à la volée *annexe 8*
- YAFFS, performant et simple *annexe 7*
- LogFS, en développement, probablement équivalent à JFFS, sauf pour le temps de montage.

4.4 Conception

4.4.1 Besoins

Avant toute chose, nous avons définit les besoins exacts du futur système de fichiers, le but étant de l'utiliser dans l'embarqué et non sur des PC traditionnels.

- Mises à jours "déplacées" (impossible de réécrire sur une page)
- Bonne répartition des écritures (limite d'effacement à 100 000)
- "Power safe" (coupures de courants sont fréquentes dans l'embarqué)
- Correction des erreurs (avec ECC)
- Portable facilement (WinCE, Linux, autre ?)
- Gestion spéciale des fichiers de log (grand nombre de petites écritures)
- Faible empreinte mémoire

4.4.2 Outils

Adeneo utilise *Trac* (http ://trac.edgewall.com/) pour ses projets. Trac est un gestionnaire de projet disposant d'une interface à Subversion, d'un wiki, et d'une gestionnaire de ticket. Cet outil aide vraiment lors de la réalisation d'un projet, seul ou en groupe, car il regroupe tout ce qui est nécessaire. Subversion permet de gérer les modifications dans le code, le wiki permet de stoker les informations, et les tickets permettent de créer une liste de chose "A faire" et de voir l'avancement global du projet.

La première version du système du fichier, pour des raisons de facilité et rapidité de développement, a été réalisée sous GNU/Linux. Pour cela, j'ai utilisé VMWare (solution de virtualisation) et Kubuntu (distribution GNU/Linux). Les principaux logiciels utilisés étant Emacs, gdb, valgrind, bash, gcc et svn.

4.5 NandFS

A cause d'un manque total d'imagination, le système de fichiers à été nommé NandFS (NAND Filesystem).

4.5.1 Principe

Comme la pluspart des systèmes de fichiers pour NAND Flash, NandFS est un "Log-structured file system" (voir *annexe 9*). Comme il n'est pas possible de réécrire deux fois de suite au même endroit, chaque modification est versionnée. Lors du montage du système de fichiers, la NAND est scannée, et la dernière version valide de chaque "objet" (fichier, répertoire) est conservée. Cela nécessite de créer en mémoire une structure représentant l'ensemble du système de fichiers.

Il y'a donc deux représentations distinctes du FS, la représentation physique (sur la NAND) et la représentation logique (en RAM). Le reste découle directement de ces deux représentations.

Le principe de base est assez simple : chaque élément présent sur le FS (répertoire, fichier, lien, socket) est appelé objet. Chaque objet possède un identifiant unique, un numéro de version et un parent. Comme il n'est pas possible de modifier une page, un objet est considéré comme valide quand son numéro de version est le plus élevé. Il existe un type d'objet particulier : les fichiers. En effet les fichiers contiennent des données, ces données sont divisées en bout de donnée ("chunk") de la taille d'une page. Chaque chunk partage sa version et son identifiant avec le fichier au quel il est associé, et possède en plus un "offset" qui représente sa position dans le fichier.

Lexique

– Object : Fichiers, répertoires, liens, sockets

– Header : métadonnées associées à un objet
– Chunk : Bout de données

Pages logiques

Comme vu précédemment, il existe une zone de spare associée à chaque page. Pour monter le système de fichiers plus rapidement lors d'un scan complet, il a été décidé de stoker les informations nécessaires au montage dans les spares.

Il existe dans un système de fichiers deux types de données :
– Données réelles : ce qu'on trouve dans les fichiers
– Métadonnées : nom des fichiers/répertoires, droits, temps de modification

Les spares sont donc utilisés différemment selon le type de donnée, la page étant ensuite utilisée pour stoker ce qu'il y'a réellement à stoker.

	Metadonnées	Données
4 octet	ID	ID
4 octet	Version	Version
1 octet	Type	Type
1 octet	Marqueur	Marqueur
4 octet	Parent ID	Offset
2 Octet	Name Hash	Inutilisé

NandFS nécessite donc un minimum de 16 octet disponible dans la zone de spare. Pour pouvoir fonctionner sur tout les types de NAND, NandFS calcule en fonction de la place disponible dans les spares le nombre de pages physiques à regrouper pour obtenir une page logique où il y aura au moins 16 octet libres en spare. Typiquement la taille des pages logiques sera 1024 octet sur une NAND Small Page et 2048 sur une NAND Large Page.

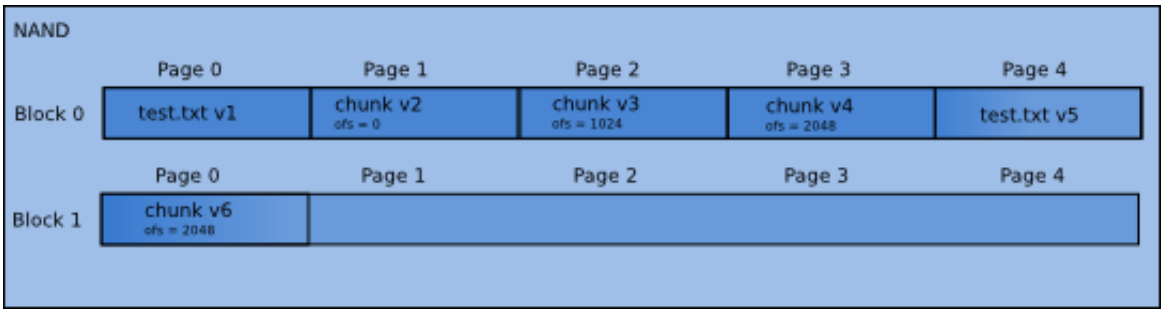
Scan et Versions

Lors du montage du système de fichiers, toutes les zones de spares sont lues et la représentation en RAM est créée. Seule la dernière version de chaque objet est gardée. L'algorithme a été fait pour créer une structure cohérente dans tout les cas. Les objet dont les parents sont introuvables sont placés dans un dossier spécial (lostnfound).

Représentation en NAND

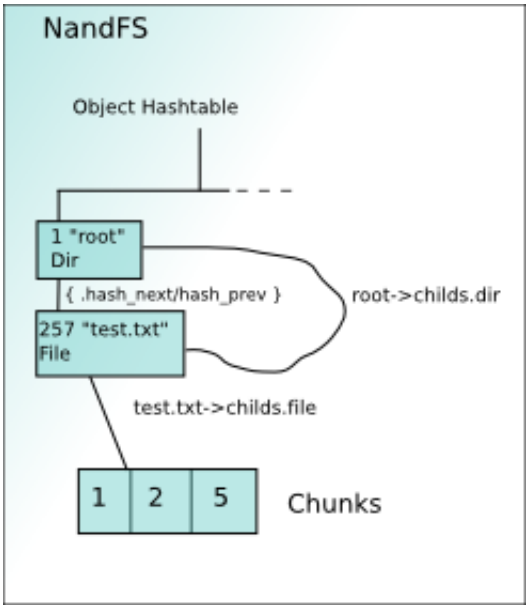
Voila un exemple de ce qui peut se trouver sur la NAND. Ici un fichier test.txt de taille ≥ 2048 a été créé, puis la fin de ce fichier a été modifiée. Il y'a donc une deuxième version des métadonnées de test.txt (nouveau temps

de modifications) ainsi qu'une nouvelle version du dernier chunk. Ainsi, les pages 0 et 4 sont maintenant obsolètes.



Représentation en RAM

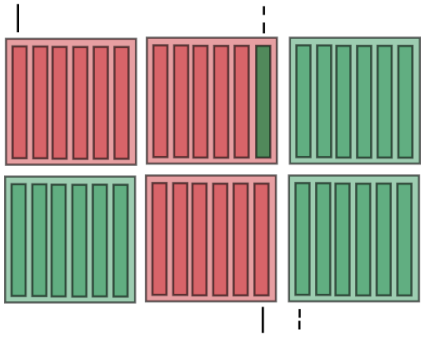
Voila la représentation en RAM du même exemple. Tout les objets sont accessible via une Hashtable dont la clef est leur identifiant unique. Les pages des chunks sont stockées dans un tableau associé a chaque fichier dont l'index est l'offset.



Allocation de page

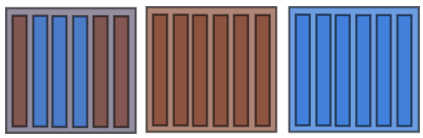
Lors de l'écriture, il faut rechercher une page libre. L'algorithme utilisé permet de garantire une bonne répartition sur la NAND (ce qui est important vu que les cycles d'écriture/effacements sont limités). Lors du scan, un tableau contenant les infos de chaque bloc d'effacement est créé (pages libres, pages obsolètes, pages illisibles). NandFS parcour se tableau, et lorsque qu'il

trouve une page libre, note le bloc de cette page comme bloc d'allocation. Une fois que le bloc d'allocation courant ne possède plus de page libre, un autre parcour est effectué à partir du dernier point d'arrêt. Le garbage collector possède deux blocs réservés : son propre bloc d'allocation, ainsi que le premier bloc libre trouvé lors du parcour du tableau.



Garbage collector

Le garbage collector permet d'effacer les blocs contenant uniquement ou un certain ratio (paramétrable) de pages obsolètes. Il peut être déclenché par un thread, ou lorsque la recherche de page libre échoue. Lorsque un bloc contient encore des pages utiles, elles sont recopiées sur un nouveau bloc (en utilisant les blocs libres réservés au garbage collector), puis rendues obsolètes.



Dans cet exemple, les pages marron sont obsolètes. Ici, lors du passage du garbage collector le bloc complètement obsolète sera effacé, libérant de la place pour copier les pages utiles du premier bloc afin de pouvoir lui aussi l'effacer.

Bad Blocks

Si un ECC est incorrect, ou s'il est impossible de lire une page, le nombre de page illisibles du bloc est incrémenté. Lors du déclenchement du garbage collector, les pages encore lisibles sont recopiées, puis les blocs sont physiquement marquées comme mauvais.

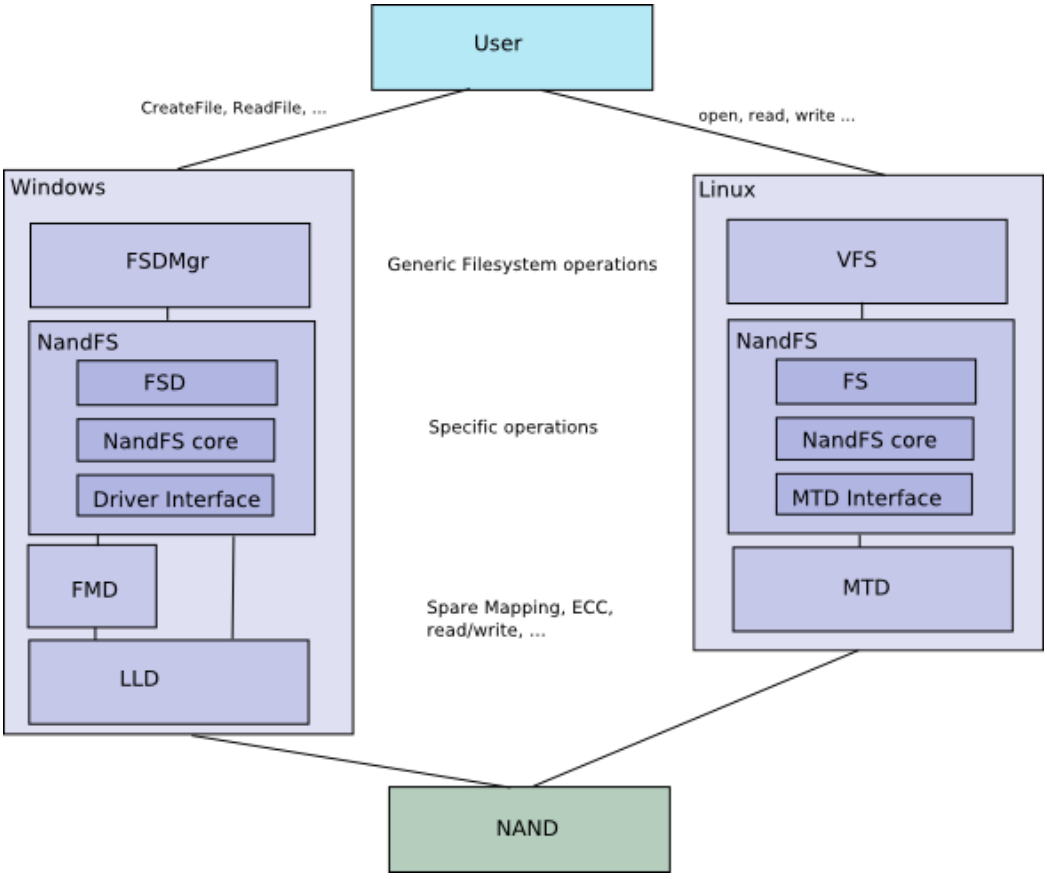
Cache

Afin de minimiser les écritures, les données ne sont pas directement écrites sur la NAND. Il existe deux caches (leader et chunks). Ces caches sont physiquements écrits lorsque :

– ils sont pleins.
– nandfs_flush_file() est appelé sur un objet.
– nandfs_sync() est appelé.
– le fichier n'a pas été modifié depuis 5 secondes.

4.5.2 Portage

Ce schéma représente l'architecture d'un système de fichiers sous Windows et sous Linux.



Il y a donc pour chaque système deux interfaces spécifiques à créer, une pour pour le côté "Filesystem" et l'autre pour le côté NAND. Il y'a en plus tout ce qui est compilation (types spécifiques, includes), et certaines fonctions spécifiques (allocation mémoire).

NandFS à été pensé dès le départ pour être indépendant du système utilisé, et facilement adaptable. Ainsi, plusieurs portages ont été réalisés.

4.5.3 Portage Userspace

User space = Espace utilisateur, en gros, c'est un programme normal

Pour pouvoir utiliser facilement des outils de débogage tels que gdb ou valgrind, un émulateur de NAND user-space et une couche d'abstraction correspondante ont été créés. L'émulateur alloue de la mémoire RAM et s'en sert comme si c'était une NAND, il permet de sauvegarder ces données pour les recharger plus tard.

Cela à vraiment été très utile pour réparer certains bugs et vérifier qu'il n'y avait pas de fuite mémoire.

4.5.4 Portage Linux

Le portage Linux se divise en trois partie :

Adaptation VFS

Sous linux, c'est le VFS qui gère toutes les opérations génériques, et appelles les fonctions du FS pour certaines opérations spécifique. Cette partie consiste donc à créer les fonctions pour ces opérations spécifiques (par exemple, lire dans un fichier, créer un répertoire, etc ..). Dans cette partie, quelques "bonus" ont été rajoutés, tels que des fichiers permettant d'afficher les statistiques du système de fichiers (dans /proc).

Adaptation MTD

La MTD est sous linux une couche générique permettant d'utiliser de la mémoire Flash. Cette adaptation a été rapide à réaliser, l'API MTD étant simple à utiliser en plus d'être très complète. En effet, cette couche permet de placer automatiquement des données dans les spares, et gère directement l'ECC.

Allocation mémoire

Linux disposant de caches d'allocation mémoire, une couche d'adaptation à été réalisé pour l'utiliser. En effet un appel à "malloc" est assez coûteux. Sous les systèmes ne disposant pas de ce types de cache, il sera nécessaire de les créer dans la couche d'adaptation.

4.5.5 Portage WinCE

Par manque de temps le portage WinCE n'a pas pu être fait. Il ressemble un peut au portage Linux (partie FS, partie Driver, partie Mémoire), mais les fonctionalité génériques (gestion des droits, gestion des flags, etc ..) ne sont pas gérées par un VFS. De plus, il n'existe pas de driver générique de mémoire Flash comme sous linux. Enfin, aucun cache d'allocation mémoire n'est prévu.

4.5.6 Empreinte mémoire

Lors de la conceptions, plusieurs changements ont été faits pour réduire l'empreinte mémoire. En effet, au début, NandFS utilisait des arbres équilibrés pour stoker les pages des chunks. Mais cela consommait bien trop de mémoire (jusqu'a 32Mo avec une NAND de 2Go pleine). Pour cette raison, des tableaux ont ensuite été utilisés.

4.5.7 Tests du système de fichier

Au cours du développement, il a fallu tester la robustesse du système de fichier. Pour cela, en plus de tests basiques (copie de fichier, démontage, remontage), des benchmarks ont été utilisés. La pluspart du temps l'utilisation de ces benchmark à été bénéfique et a permis de trouver et corriger de nouveaux bugs. De plus, des tests de coupures de courants ont été réalisés, pour vérifier que l'algorithme de Scan de la NAND permettait bien de recréer une structure cohérente.

4.5.8 Performances

Une fois le système de fichiersstabilisé, des tests de performances ont été réalisés sur les cartes. Peut de test ont été faits avec des systèmes de fichier tels que FAT et ext2, les performances de ceux-ci étant de toute façon moins bonnes.

Tests basiques

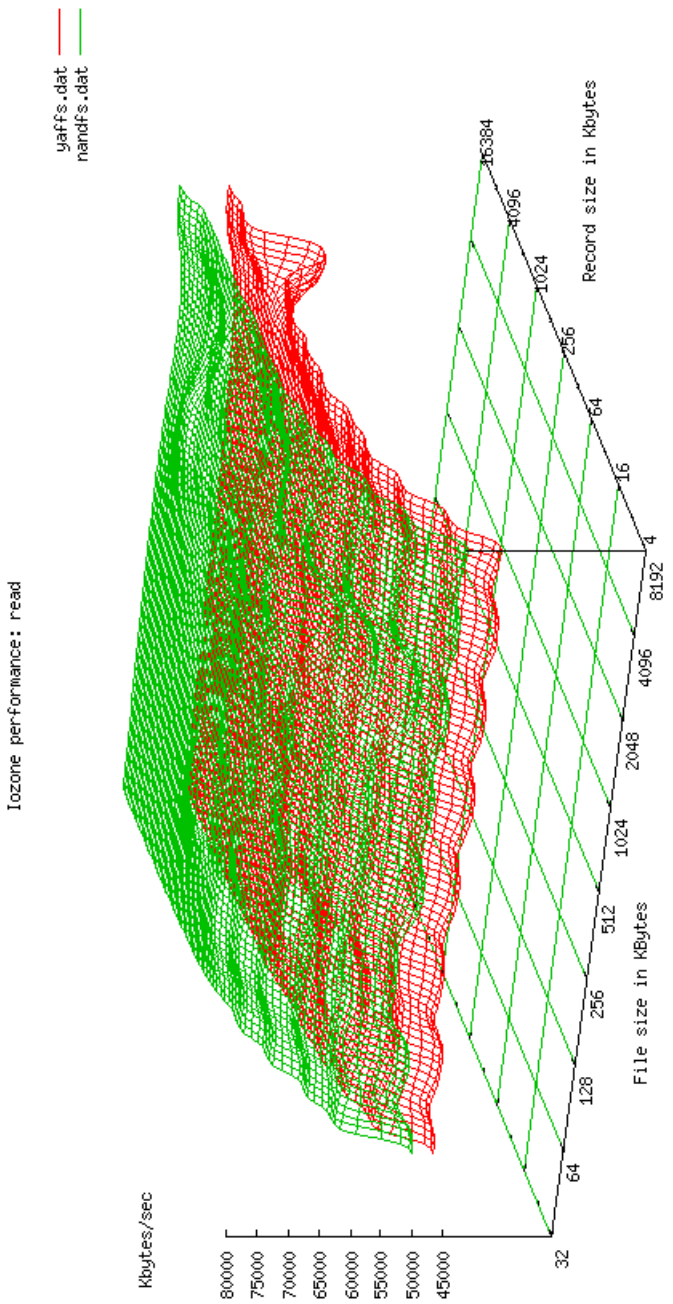
Copie de 500 fichiers (pour une taille totale de 66Mo) :
- YAFFS : 26 sec
- JFFS : 2m39 sec
- NandFS : 25 sec

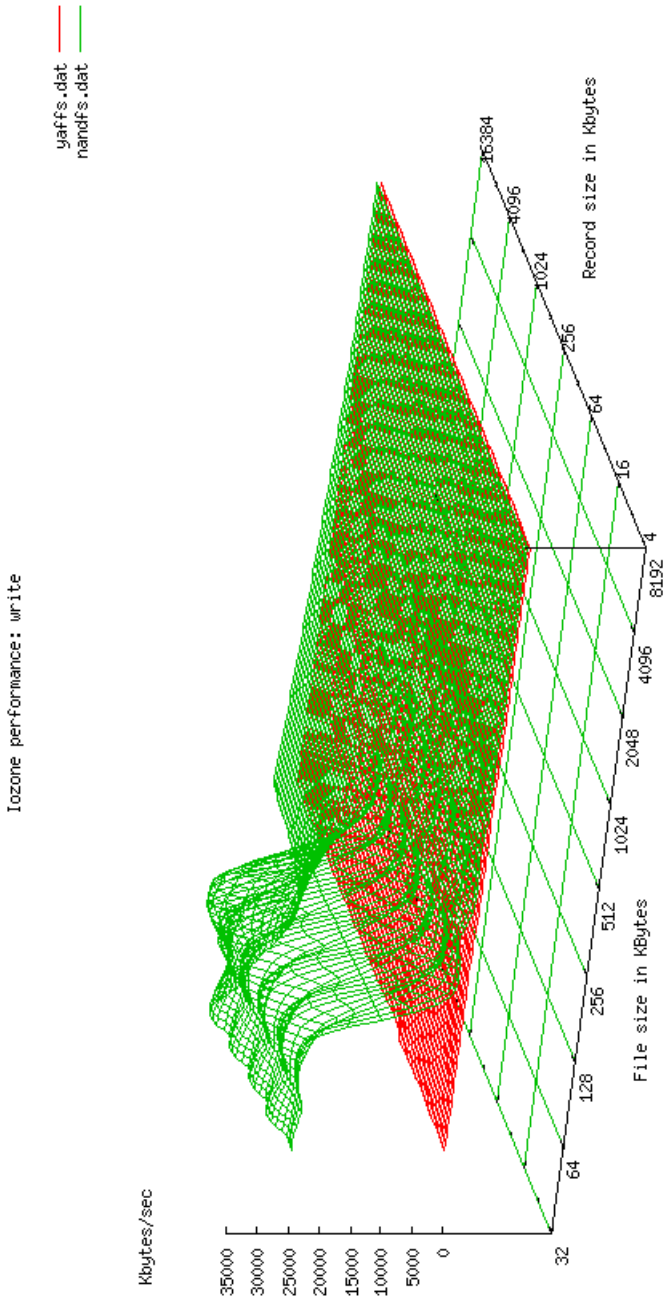
Copie de 16 Mo
- YAFFS : 6 sec
- JFFS : 18 sec
- NandFS : 6 sec

Iozone

Tests réalisés sur AT91SAM9263 avec une NAND 8 Bit
Résultats complets en *annexe 10*

On remarque qu'en lecture les performances sont très élevées, cela est du aux divers caches dont dispose Linux. La différence entre les deux systèmes de fichiers est par contre significative. En écriture, NandFS est très performant lors des petites opérations grâce à des caches internes. Le reste du temps les performances sont uniquement limitées par la NAND.


Iozone performance: read
yaffs.dat
nandfs.dat
Kbytes/sec
80000
75000
70000
65000
60000
55000
50000
45000
File size in KBytes
32
64
128
256
512
1024
2048
4096
8192
Record size in Kbytes
4
16
64
256
1024
4096
16384



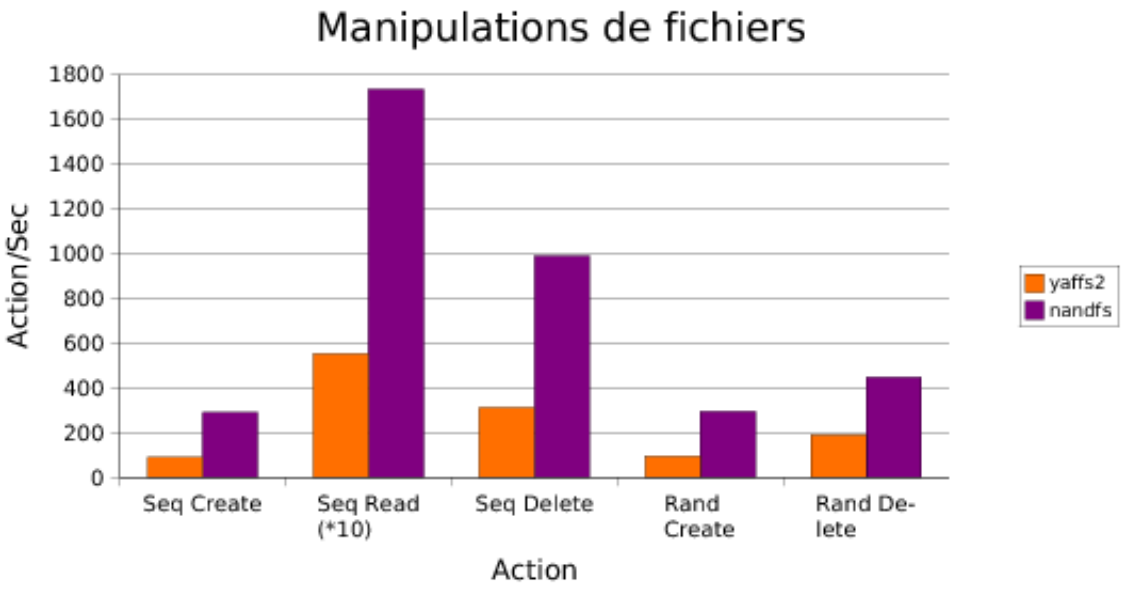
Linux disposant d'un cache au niveau VFS, des tests avec des fichiers dont la taille dépasse la taille de la RAM ont été faits :

`YAFFS`

```
              KB  reclen   write rewrite    read    reread
           60000       4    2055    1929    3851      3864
NandFS
              KB  reclen   write rewrite    read    reread
           60000       4    2075    2061    4181      4196
```

bonnie++

Tests réalisés sur AT91SAM9261 avec une NAND 16 Bit



Temps de montage

Tests réalisés sur AT91SAM9261 avec une NAND 16 Bit
Résultats complets en *annexe 10*

	NandFS	YAFFS	JFFS
NAND “Propre”	0.18 sec	0.20 sec	1.28 sec
NAND “Sale”	2.66 sec	0.13 sec	9.95 sec

- Propre : la NAND n’a jamais été utilisée
- Sale : toutes les pages de la NAND ont été utilisées au moins une fois

Nombre d’écritures

Tests réalisés avec nandsim (simulateur de NAND) Sur une NAND flash, il faut veiller à réduire au maximum le nombre d’écritures à cause de la durée de vie des blocs d’effacement.

Nombre de pages écrites par opération

	NandFS	YAFFS
Création fichier	1	2
Création répertoire	1	2
Écriture de 64K	2	2
Ajout de 64K	2	2
Suppression du fichier	1	1
Suppression du répertoire	1	2

Note : entre chaque opération, attente de 5 secondes, sinon dans NandFS tout se passerais dans les cache et au final seul deux pages seraient écrites (header + données).

Conclusions

Le seul système de fichierscomparable est YAFFS2. JFFS2 est bien plus lent, et prend plus de mémoires, et LogFS n'est pas encore assez stable pour être utilisé.

Les performances en lecture/écriture sont comparables à celles de YAFFS2, et en général sont limitées par la NAND. Les performances en terme de manipulation de structure (création/suppression de fichiers/répertoire) sont jusqu'a trois fois plus rapide avec NandFS qu'avec YAFFS2. Les performances en terme de nombre d'écritures sont meilleures sous NandFS grâce à l'utilisation des caches internes. Les performances en terme de temps de montage sont meilleures sous YAFFS2, mais l'implémentation d'un système de checkpoint dans NandFS devrais permettre de palier à ce problème. L'empreinte mémoire est compliquée à mesurer, mais elle devrais théoriquement être plus faible sous NandFS que sous YAFFS2.

4.6 Présentation

Durant mon stage, j'ai eu l'occasion de présenter mon sujet aux autres membres du département logiciel, cette présentation est disponible dans les annexes.

4.7 Documentation

Pendant toute la durée du stage, j'ai créé et maintenu la documentation relative à NandFS. Cette documentation est divisée en deux partie, la première contenant les spécifications de NandFS, l'autre les informations pratiques pour utiliser NandFS sous Linux. Cette documentation a été réalisée sur le wiki et est partiellement disponible dans les annexes.

Chapitre 5

Conclusion

Le stage s'est déroulé dans une ambiance de travail conviviale. En effet, ADENEO est une société dynamique où tout est mis en oeuvre pour permettre une bonne entente entre les salariés.

Ce stage m'a permis de découvrir le monde de l'entreprise. J'ai aussi eu l'occasion d'accroître mes connaissances dans le monde de l'embarqué et celui de la programmation système. Le sujet de ce stage était particulièrement intéressant et assez peu abordé.