Microlinux : Solution de virtualisation des postes de travail

Nicolas Hordé, Eric Trezel, Stephane Rocher

Microlinux : Solution de virtualisation des postes de travail

Nicolas Hordé
D.S.I
Faculté des Sciences et Techniques
123 Avenue Albert Thomas
87060 Limoges CEDEX

Stéphane Rocher
D.S.I.
123 Av. Albert Thomas,
87060 Limoges CEDEX

Eric Trézel
D.S.I
Faculté des Sciences et Techniques
123 Avenue Albert Thomas
87060 Limoges CEDEX

Résumé

Notre solution permet de cloner des machines virtuelles (VM) depuis un serveur central vers un ensemble de clients en respectant des critères définis préalablement à partir d'une interface dédiée.

Microlinux comprend :

- *un serveur (Ubuntu 14.04.2LTS):*
  - *une base de données contenant les informations de déploiement des VMs (PostgreSQL)*
  - *plusieurs scripts permettant le déploiement multicast des VMs (Python, Bash)*
  - *un système de démarrage/installation PXE du Microlinux (TFTP/HTTP)*
- *une interface d'administration codée en Jython[1], elle s'exécute dans une machine Java ; multi-plateforme :*
  - *Les administrateurs choisissent ici les différents critères qui discriminent les ordinateurs sur lesquels les machines virtuelles seront poussées ainsi que les usagers ayant l'autorisation de les utiliser.*
- *un hyperviseur oVirt permettant de réaliser la préparation des VMs par les usagers de la solution (enseignants).*
- *des clients basés sur Debian Jessie, bootable depuis le réseau (PXE) ou depuis le disque après une installation préalable :*
  - *Les usagers finaux (étudiants, administratifs) visualisent une interface (WEB/HTTP) sur leur poste de travail générée à partir des différents critères choisis par les administrateurs. Il leur est possible d'accéder à des machines virtuelles, des connexions RDP/SPICE ou encore des pages Web.*

1. Interpréteur Python écrit en Java. Portable, il offre la possibilité de manipuler des classes Java.

Cette solution permet de faire cohabiter différents systèmes d'exploitation sur un même poste et de procéder à la réinstallation par lot même si les usagers travaillent sur une autre machine virtuelle. Il est ainsi plus aisé de procéder à la réinstallation des postes de travail, dynamisant la gestion du parc informatique et banalisant de ce fait des salles qui étaient jusqu'alors dédiées.

D'autre part il devient possible d'exécuter plusieurs systèmes d'exploitation simultanément (TP "réseaux").

Mots-clefs

VM, OVIRT, Virtualisation, Linux, RDP, SPICE, TFTP, HTTP, PostgreSQL, Python, Jython, KVM, multicast

1 Infrastructure de la solution

1.1 Vue d'ensemble

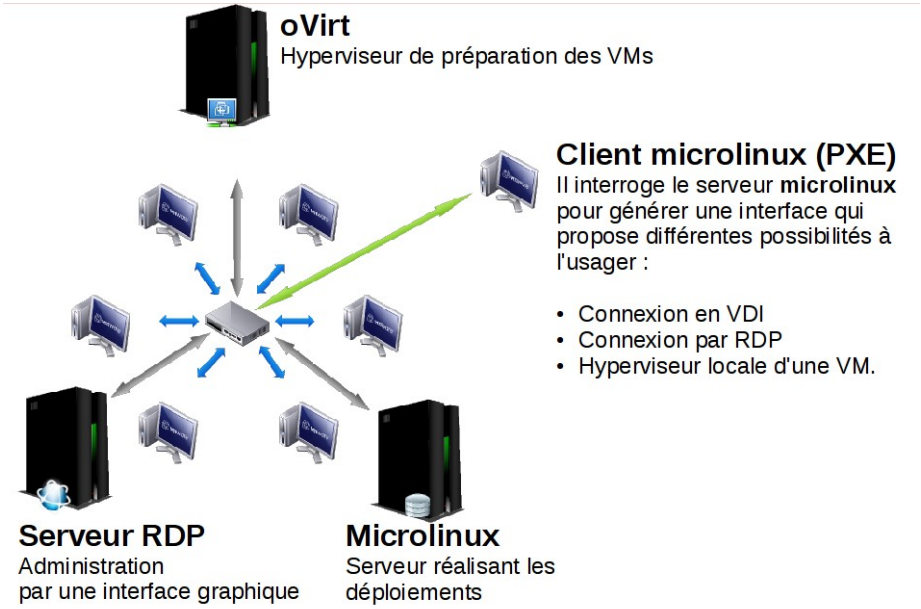

Illustration 1: Schéma d'ensemble de la solution Microlinux.

La solution Microlinux est basée sur un ensemble de briques du monde du logiciel libre, et comme l'indique le nom, majoritairement sur Linux.

L'ensemble de la solution inter-opère à partir du réseau sur un modèle client-serveur. Les transactions s'appuient sur le protocole IP qui rend la solution capable de fonctionner à travers différents réseaux .

La structure de la solution Microlinux a été imaginée afin de s'intégrer au mieux au sein dans notre infrastructure existante :

- ainsi la console d'administration de Microlinux s'est logée sur notre serveur SVP-DSI qui centralise la majorité des applicatifs liés au support. Elle est donc accessible par une session de travail distant en RDP ;
- le serveur Microlinux est sous forme d'une machine virtuelle qui est hébergée sur notre

infrastructure VMWare. Ce serveur héberge la base de donnée qui comprend l'ensemble des informations permettant d'automatiser le clonage des machines virtuelles depuis l' « hyperviseur de préparation des VMs » vers les clients. De nombreux scripts en Bash et en Python constituent le « backend » de la solution Microlinux. C'est ce serveur qui fait le lien entre les paramètres saisis sur l'interface d'administration, le backend de clonage multicast et l'hyperviseur de préparation des VMs : C'est l' « intelligence » du système ;

- l'hyperviseur de préparation des VMs n'est autre qu'un hyperviseur oVirt tout à fait classique autour duquel nous avons greffé d'autres composants pour composer la solution. C'est à partir du front-end de oVirt que les utilisateurs (enseignants) vont préparer leurs machines virtuelles qui seront ensuite clonées en multicast vers les différents espaces pédagogiques. Et ce, selon les critères et la programmation horaire renseignés préalablement par un administrateur de la solution à partir de la console d'administration ;
- les clients sont des ordinateurs de travail traditionnels que l'on pourrait qualifier de « clients lourds ». Ils sont « convertis » en client Microlinux par l'installation d'un micro-système basé sur Linux (d'où le nom) ou en démarrant directement depuis le réseau sur le serveur Microlinux. Cette micro distribution embarque un hyperviseur KVM, d'un client RDP, d'un client SPICE et d'un navigateur web qui constitue d'ailleurs l'interface du client. Il devient donc possible à partir de ce client de démarrer une session de travail par l'hyperviseur locale ou d'initier une session distante en RDP/SPICE ou HTTP/WEB.

1.2 Fonctionnalités

Les fonctionnalités du système corroborent le cahier de charge que l'on a dressé avant la conception du système, à savoir :

- clonage de machines virtuelles par lot depuis une infrastructure centralisée vers différents clients léger ou lourd ;
- console d'administration centralisée autorisant la gestion des droits des machines virtuelles avec une granularité s'appuyant sur les ordinateurs déclarés dans le DNS (Netmagis, une application de gestion DNS/DHCP) et sur les usagers renseignés dans notre annuaire LDAP ;
- un client pouvant être amorcé de façon locale ou distante, et qui pourrait fonctionner en tant qu'hyperviseur local et ce, déconnecté du réseau (suite à une panne par exemple, ou en isolement volontaire).

1.3 Considérations réseaux

Une grande partie du système se base sur la diffusion de donnée en multicast et sur le réveil distant des ordinateurs, ce qui n'est pas sans poser certaines difficultés :

- réveil et démarrage d'ordinateur impossible suite au positionnement des ordinateurs dans un état d'alimentation impropre à l'usage de la technologie « wake on lan » (extinction « sauvage ») ;
- propagation des données impossibles du serveur vers l(es)'hôte(s) à cause d'ACL ou de pare-feu, ou réglage des stratégies multicast. Il est parfois difficile d'opérer une politique de filtrage centralisée sur plusieurs campus...

2 Serveur

2.1 La base de donnée

Pour des raisons pratiques, essentiellement dues au fait que Netmagis, une partie de notre infrastructure sur laquelle se base la gestion DNS, exploite une base de donnée en Postgresl, nous avons choisi cet « object-relational database management system ». Le cadre de ce projet étant la pédagogie avec une charge supposée raisonnable nous n'avons pas intégré de load balancing permettant d'améliorer la disponibilité et/ou la sécurité des informations. De plus la base de données n'est accédée qu'à partir de la console d'administration et du « backend » de gestion des VMs.

2.2 Le « Back-end »

Cette partie est le cœur du système. Nous avons développé un ensemble de scripts en python dont l'exécution est planifiée par « cron ». Ils interrogent la base de données régulièrement afin de connaître les opérations d'envoi multicast à réaliser. Avant de procéder à l'envoi multicast des VMs par le biais de udp-sender (Udpcast), le script sollicite oVirt par son API REST afin d'exporter la VM vers un partage NFS. La VM comprend les données binaires des systèmes de fichiers utilisés par la machine virtuelle, ainsi que le descriptif de la machine virtuelle. Le script procède ensuite méthodiquement à l'envoi des systèmes de fichiers vers les différents clients avec un flux multicast unique :

- il réveille les clients par le biais d'une trame WOL ;
- les clients démarrent en PXE ou sur le disque local, le système Microlinux ;
- le back-end vérifie que tous les clients sont opérationnels ou notifie le manque de certains clients ;
- il procède à la préparation des clients à la réception de données par le biais de connexions SSH concourantes (lancement de Udp-receive) ;
- il démarre le processus d'envoi côté serveur (lancement de udp-sender) ;
- chaque client peut procéder à une vérification de contenu par hash (optionnel) ;
- le serveur notifie l'administrateur ayant programmé le clonage que l'opération a été réalisée avec un compte rendu détaillé des opérations, des co-destinataires peuvent être informés également.

2.3 Amorçage du client depuis le réseau

Afin de réaliser l'amorçage distant des clients « Microlinux » ou encore leur installation. Le serveur Microlinux est doté d'un serveur TFTP/HTTP. Le client Microlinux est présenté aux clients sous forme d'un noyau et d'un INITRD, ainsi que d'une arborescence Linux complète compressée (squashfs compressé en XZ). Ainsi voici le processus de démarrage :

- le client démarre et sollicite le serveur DHCP (notre serveur universitaire) ;
- celui-ci lui attribue une IP, et lui renvoie les informations nécessaires (nom du serveur, nom de l'image) à un amorçage par PXE (depuis le serveur Microlinux) ;
- le client charge en TFTP le noyau linux et l'image de démarrage (depuis le serveur Microlinux) en mémoire et l'exécute ;
- puis conformément à la ligne de commande spécifiée précédemment, le noyau procède au chargement en *ram* de l'arborescence compressée par le biais du protocole HTTP (toujours depuis le serveur Microlinux) ;
- l'arborescence permet au noyau de réaliser un démarrage complet de la distribution Linux qui lance un session X, puis un gestionnaire de fenêtre Openbox ;
- une interface graphique basée sur un client Web, comportant un cache local permettant une navigation offline, donne à l'usager la possibilité de lancer différentes tâches : authentification, installation locale de microlinux (pour les administrateurs logués), initiation d'une session RDP/SPICE, lancement d'une VM en hypervision locale, etc.

3 Interface d'administration

3.1 Présentation de l'interface

L'interface devait répondre à des critères de portabilité en cas de changement de système d'exploitation de notre serveur applicatif. Le Jython qui est un langage de script issu de Python s'appuyant sur une machine virtuelle JAVA, est le langage qui a été retenu pour réaliser cette interface. Cette interface devait accéder en postgresql à la base de données de Microlinux mais aussi à l'hyperviseur oVirt par le biais de l'API REST, ainsi qu'à l'annuaire LDAP et à Netmagis (gestion DNS) : seuls JAVA et Python satisfaisaient

honorablement à ces conditions. Ce choix s'est opéré par sa simplicité de développement en utilisant une interface prodigué par les API de Swing tout en conservant la flexibilité d'un langage de script.

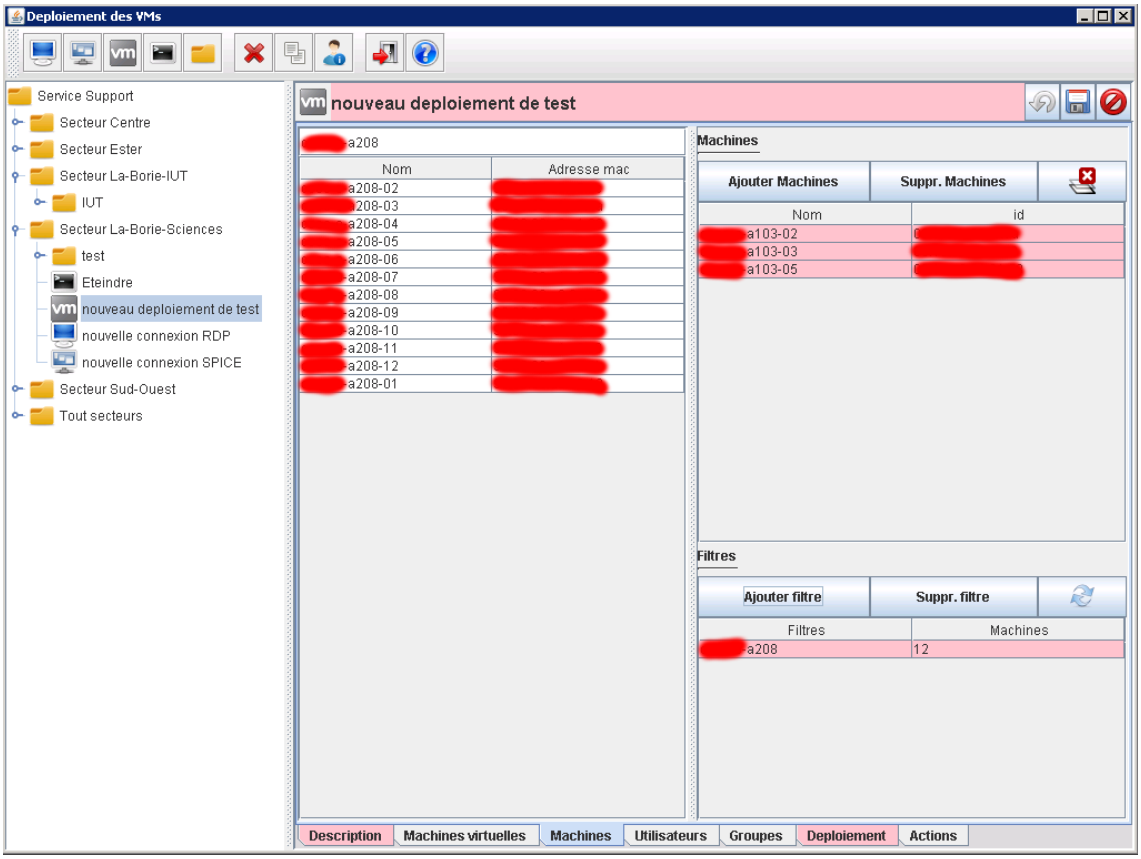

Illustration 2: Console d'administration, gestion des "machines".

3.2 A venir…

L'interface dans sa version finale devraient permettre de lancer des tâches par lot sur les clients attachés à une tâche de déploiement, tel que :

- démarrage des clients ;
- lancement et extinction d'une machine virtuelle ;
- clonage instantanée d'une machine virtuelle depuis l'hyperviseur de préparation ;
- vérification de la connectivité réseau vers les clients ;
- versionning et test d'intégrité des disques virtuels déployés sur le client ;
- ajout d'un fichier à l'intérieur du système de fichier d'une VM, etc.

4 Hyperviseur de préparation des VMs

Ovirt comporte 2 volets différents accessibles par le biais d'une interface web :

- la partie « usagers » qui permet d'effectuer des tâches simples parmi lesquelles figurent le démarrage et la préparation d'une machine virtuelle ;
- la partie administrateur qui permet d'effectuer des tâches étendues de gestion tel que l'affectation de cartes réseaux vers des VLAN spécifiques ou encore la création d'instantanées.

4.1 Partie usagers

Cette partie est dédiée aux usagers, qui sont dans notre cas, les enseignants qui souhaitent préparer des machines virtuelles afin de dispenser des cours au sein de nos espaces pédagogiques. L'authentification des usagers se base sur l'annuaire commun de l'université (ici AD depuis LDAP), et, lorsque l'enseignant se connecte, il bénéficie d'une machine virtuelle mise à sa disposition par un administrateur avec un système de base correspondant à ses attentes. À lui de personnaliser sa machine virtuelle tout en respectant le cadre fixé par la charte de déontologie de l'université, afin de créer un environnement de travail fidèle à l'enseignement visé. Dans le cas de cours orientés vers l'informatique et les technologies numériques, il paraît intéressant d'avoir cette démarche, mais dans le cas de cours plus « classique » c'est le gestionnaire de parc qui peut devenir l'usager et créer une machine virtuelle correspondant aux besoins de l'enseignant. Une fois ce travail terminé, il convient de reprendre contact avec un administrateur pour valider la machine virtuelle et programmer dans l'interface de gestion un déploiement vers les clients souhaités.

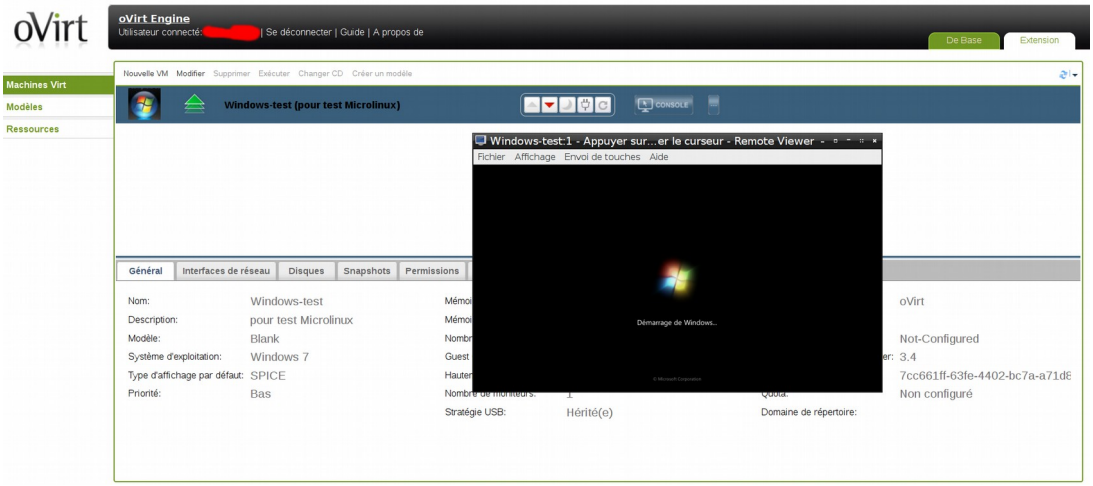

Illustration 3: oVirt, console usagers.

4.2 Partie administrateur

Seuls les administrateurs ont accès à ce volet de l'hyperviseur afin d'instancier des « templates » (modèles) et affecter des droits aux usagers. Il est aussi possible de réaliser des tâches avancées d'administration comme la migration à chaud de VMs en cas de maintenance d'un hyperviseur, ou encore une affectation de VLAN sur une carte réseau d'une VM afin que celui-ci corresponde à celui de l'espace pédagogique où la VM fonctionnera ! Bref tout ce que permet un hyperviseur classique, car aucune modification n'a été apportée à cette brique de la solution Microlinux.

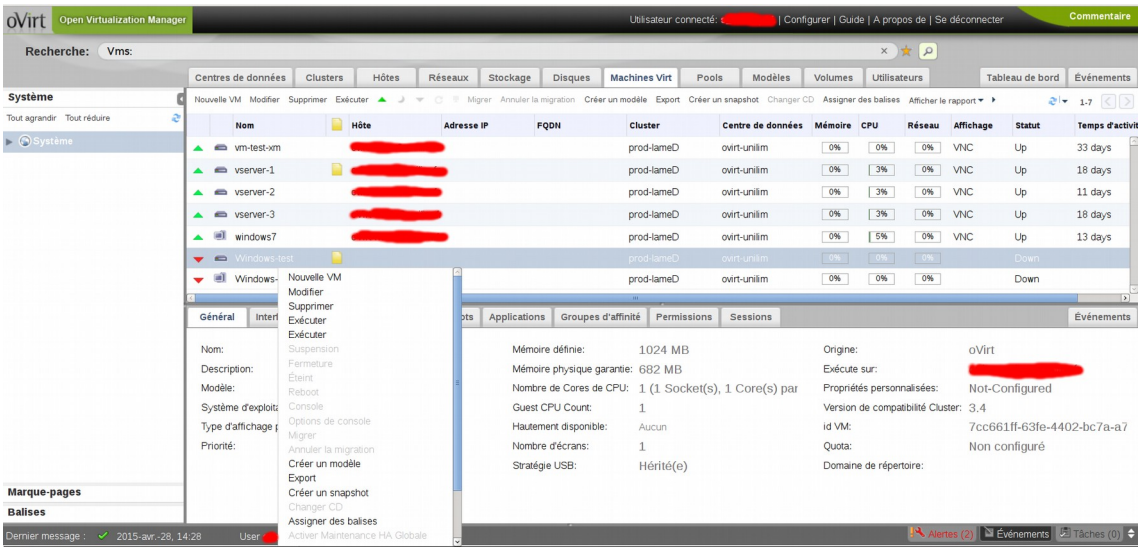

Illustration 4: oVirt, l'interface administrateur.

Le client

4.3 Microlinux à proprement parler

Le client est basé sur une installation minimale de Debian Jessie stable sur laquelle des paquets sont installés tels que :

- qemu-kvm qui est l'hyperviseur local,
- python qui est le langage de script qui anime la solution,
- spice-client est le client spice,
- rdesktop permet d'initier des connexions RDP,
- xorg qui est le serveur graphique,
- openbox gère le fenêtrage,
- udpcast opère le transfert multicast des images de disques virtuels,
- lzop est le logiciel de compression qui permet de réduire à la volée (avec un bon ratio temps/compression) les images de disques virtuels lors de l'envoi RDP,
- ...un nombre assez important de paquet.

Se rajoute à ce client une interface dynamique développée sur mesure qui se nomme « light ».

L'ensemble des fichiers qui constituent la distribution est ensuite archivé sous forme d'un fichier squashfs (un système de fichier compressé en lecture seule) compressé en XZ afin de permettre un envoi efficient par HTTP dans l'optique d'autoriser un amorçage par le réseau.

Pour exploiter le client en RDP/SPICE, aucune installation n'est nécessaire… Il suffit de démarrer par le réseau et initier en un clic une connexion vers votre serveur de travail. Cet usage peut donc être réalisé à partir d'ordinateurs diskless.

Cependant pour utiliser de l'hypervision locale, il est indispensable (pour le moment) de procéder à la

réservation d'un espace disque afin de stocker les images de disques virtuels. Pour ce faire un outil permet la création ou le formatage d'une partition dans un format BTRFS (Butter FS). Sur cette partition, il est possible d'installer un système Microlinux complet pour autoriser un démarrage local.

4.4 L'interface dynamique

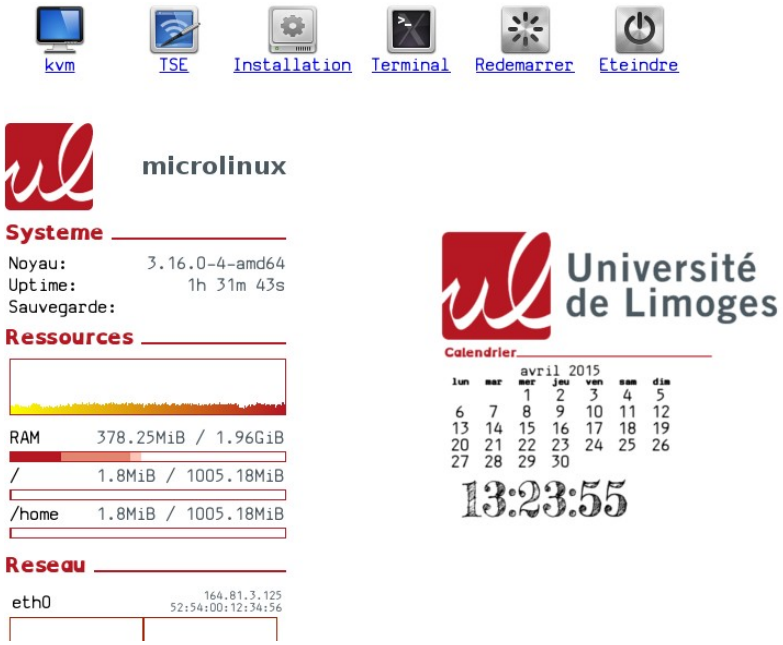

Illustration 5: Interface du client Microlinux (en cours de développement).

Dénommée « Light », cette interface a été développée en python à partir du Webkit GTK. Il s'agit en fait d'un navigateur plein écran qui s'affiche en mode « bureau ». Lorsque l'interface se lance, si une connexion est possible vers le serveur Microlinux en HTTP, il l'initie et récupère la dernière version de l'interface. Dans le cas contraire il utilise un cache local à partir de sa partition réservée, afin de générer l'interface. Ce système permet d'assurer d'avoir une version à jour et cependant autorise un fonctionnement non connecté (offline).

Il est possible d'envoyer du contenu riche vers cette interface telle que de vidéos ou des animations car l'ensemble de l'interface est réalisée à partir de page web dynamiques. Le bureau « light » est donc composé de la partie cliente/cache et de la partie page web qui est stockée au sein de serveur microlinux. Les pages utilisant les technologies PHP/Postgresql s'adaptent en fonction des ordinateurs clients et des usagers authentifiés (ou non).