Institut Supérieur d'Informatique, de
Modélisation et de leurs Applications

24, Avenue des Landais
BP 10 125
63173 Aubière cedex

Organisation Européenne
pour la Recherche Nucléaire

CERN, 385, Route de Meyrin
1217 Meyrin
Suisse

Rapport d'ingénieur

Stage de 2ème année

Filière 5 : Réseaux et télécommunications

Réaliser un programme puis une interface graphique unifiant et automatisant l'accès aux consoles des serveurs réels ou virtuels

Présenté par : SCOTT Alexandre

Responsable ISIMA : MESNARD Emmanuel

Responsable entreprise : BRARDA Loïc

4 septembre 2013

Institut Supérieur d'Informatique, de
Modélisation et de leurs Applications

24, Avenue des Landais
BP 10 125
63173 Aubière cedex

Organisation Européenne
pour la Recherche Nucléaire

CERN, 385, Route de Meyrin
1217 Meyrin
Suisse

Rapport d'ingénieur

Stage de 2ème année

Filière 5 : Réseaux et télécommunications

Réaliser un programme puis une interface graphique unifiant et automatisant l'accès aux consoles des serveurs réels ou virtuels

Présenté par : SCOTT Alexandre

Responsable ISIMA : MESNARD Emmanuel

Responsable entreprise : BRARDA Loïc

4 septembre 2013

Remerciements

Tout d'abord je tiens à chaleureusement remercier Loïc Brarda, mon maître de stage, pour le soutien, l'aide, et les nombreuses explications qu'il m'a apportés tout au long de ce stage. Mohamed Chebbi, qui est l'autre administrateur système dont j'ai envahi le bureau, m'a également beaucoup aidé, je le remercie donc beaucoup.

Je tiens également à remercier Niko Neufeld pour ses explications sur la physique en jeu au sein du CERN, et les anciens isimaliens Christophe Hoen et Pierre Schweiter (qui étaient là pour quelques semaines) pour leurs explications.

De manière générale je suis reconnaissant à l'ensemble des membres du département Online du LHCb pour leur accueil, et pour leur bonne humeur qui ont rendu mon stage très agréable.

Enfin, un grand merci à Monsieur Mesnard pour sa prévenance.

Acknowledgement

Firstly I would like to thank warmly my internship supervisor, Loïc Brarda for his support, his constant help, and his numerous explanations throughout my entire internship. I shared an office with him and another system administrator, Mohamed Chebbi, whom I also thank very much for his kind and cheerful help.

I would also like to thank Niko Neufeld for his explanations of the physics of CERN, and the former students of ISIMA Christopher Hoen and Pierre Schweitzer for their help and explanations.

I also wish to thank the entire online team of the LHCb for their welcome, and for their good team spirit which made my internship it very satisfying.

Finally I would like to thank Emmanuel Mesnard (my ISIMA mentor) for his thoughtfulness.

Résumé

Le CERN est une grande organisation par la taille qui a besoin d'énormément de ressource informatique. Pour la section où je me trouve, c'est près de 3000 serveurs qui sont employés.

Pour accéder à ces serveurs, même lorsqu'il y a un problème, il existe de nombreuses méthodes s'appuyant sur des technologies (comme IPMI, ipmitool, ou spice que j'expliquerai dans ce rapport) relativement nouvelles. La multiplicité des technologies et les différences à l'intérieur même d'une technologie rendent le fait de vouloir contacter un serveur en particulier périlleux.

En effet il faut d'abord déterminer la technologie utilisée par ce serveur pour ensuite essayer différentes méthodes...

Ce qui m'était demandé était de créer un programme qui unifie les accès à tous ces serveurs. Les administrateurs systèmes (car ce sont eux les bénéficiaires principaux de mon programme) ne doivent que savoir le nom du serveur pour pouvoir le contacter.

Mots-clefs : Serveurs, IPMI, réels/virtuels, Python, console

Abstract

The CERN is a big organisation with tremendous computational needs. For the department I was in, there are about 3000 servers that are used for computing.

To access these servers there are many different ways which rely on numerous and quite new tecnologies (like IPMI, ipmitool, or Spice that I will detail later). The problem is that they are too many ways to contact servers, but that any particular server cannot usually be contacted by multiple ways. You can therefore see that it could take some time to contact a particular server.

What I was asked to do was to design a program that unifies access to all the servers. The system administrators that use the program should only have to know the hostname of the server they are trying to contact, and a console redirection or a SOL connection will be established.

Keywords : Servers, IPMI, real/virtual, Python, console

Table des matières

Table des figures

Lexique

Certains éléments ne sont que rapidement décrit car ils sont expliqués plus en détail dans le contenu du rapport.

API (Application Programming interface) : c'est en fait une interface de programmation, c'est-à-dire qu'un logiciel met à disposition cette API pour que d'autres logiciels puissent facilement utiliser les différentes fonctionnalités qu'il comporte.

Bytecode : c'est un code intermédiaire, plus proche du langage machine que ne l'est le code source. Il est contenu dans un fichier binaire et permet une plus grande portabilité (le Java, le Python, ou le PHP utilise ce code).

GET (requête) : c'est une méthode de requête utilisée dans le protocole HTTP qui est le protocole utilisé pour internet. On peut envoyer des données avec cette méthode qui seront stockés dans l'URL (http://www.google.fr/#q=ISIMA par exemple ou q est une variable contenant ISIMA).

IPMI (Intelligent Platform Management Interface) : Outil intégré dans les serveurs qui permet de les gérer à distance. L'IPMI est particulièrement utile lorsqu'il y a des problèmes (typiquement au démarrage) qui empêchent la connexion au serveur lui-même.

JavaScript : c’est un langage de programmation de scripts orienté objet. Il a été créé en 1995 par Brendan Eich, et sert principalement pour coder des pages web interactives.

Java Web Start : c’est une technologie qui permet à Java de déployer ses applications directement depuis un navigateur web. Il se sert pour cela d’un fichier JNLP, vers lequel pointe une des pages du site internet.

JNLP (Java Network Launcher Protocol) : c’est l'extension du fichier qui sert la technologie Java Web Start. Il décrit l’application Java et ses dépendances.

JSON (JavaScript Object Notation) : format de sérialisations de données. Il a été créé pour le langage JavaScript qui est très utilisé par les technologies d’internet. C’est donc sans surprise que je peux vous dire qu’il est particulièrement utilisé pour transmettre des données sur les réseaux (spécialement les objets).

Module (Python) : un module en python est un fichier contenant du code Python, et qui peut être importé par un autre.

POST (requête) : c’est une méthode de requête utilisée dans le protocole HTTP qui est le protocole utilisé pour internet. La particularité de cette méthode est que les éléments transmis par le client (navigateur web dans la majorité des cas), sont cachés de l’utilisateur dans le corps de la requête.

RHEV (Red Hat Entreprise Virtualization) : c’est un outil de gestion d’un parc de machines virtuelles produit par Red Hat.

Sérialisation/Désérialisation : c’est un procédé qui permet de coder des informations sous la forme d’une suite d’information atomique. Ce procédé est couramment utilisé pour sauvegarder des informations, ou pour les transmettre sur un réseau.

SOL (Serial Over Lan) : c’est un mécanisme qui permet de transporter des paquets IP sur une liaison série.

XML (eXtensible Markup Language) : c’est le format de sérialisation le plus utilisé. Il a pour objectif de représenter des données (comme un objet, ou des préférences graphiques par exemple) dans un fichier, et de façon compréhensible.

Introduction

L’institution dans laquelle j’ai fait mon stage est le CERN. Cette organisation fut fondée par de nombreux pays européens en 1954, et a pour but la recherche physique théorique. Pour cela il fut construit une multitude d'accélérateurs de particules qui, lorsqu’ils fonctionnent produise un grand volume de données. Il faut ensuite pouvoir stocker ces données et les traiter.

De cette problématique vient la nécessité de structures informatiques très importantes et novatrice. Lors du dernier fonctionnement du LHC ce sont 15 pétaoctet de données, par année, qui ont été enregistrées. Pour que ces données soient traitées le CERN a mis en place un système de « grid computing », c’est-à-dire qu’il divise le problème en petits morceaux qui sont ensuite envoyés aux quatre coins du monde.

Mais avant il faut arriver à trier les données qui arrivent en flux tendu directement du détecteur. En effet, si aucun tri n'était fait, le CERN se retrouverait très vite débordé par la masse de données. Par ailleurs, ces données sont en grande partie inutiles. Les collisions produisent dans la très grande majorité des éléments inintéressants. Ce que cherchent les physiciens ce sont les exceptions qui se produisent très rarement (0,001% du temps).

Il faut donc des méthodes de tri très rapides pour faire la différence entre les résultats de collisions classiques et les résultats d’exceptions. Et par extension, il faut des moyens informatiques pour pouvoir les faire s’exécuter. Le service dans lequel j’ai effectué mon stage, s’appelle le service online du détecteur LHCb. C’est lui qui s’occupe des moyens informatiques qui effectuent ce travail de tri sur les données

qui sortent du LHCb. C'est près de 3000 serveurs directement ou indirectement impliqués qui constituent la puissance informatique du service.

La raison qui a poussé mon maitre de stage à recruter un stagiaire est la difficulté qu'ont les administrateurs systèmes à contacter des serveurs ayant des problèmes dûs aux nombreuses méthodes qui existent pour les différents types et marques différents de serveurs. Il m'a été demandé d'écrire un programme pour unifier l'accès à ces serveurs. L'utilisateur ne devra plus que connaître le nom du serveur et la connexion devra être faite de manière automatique.

Il n'existe pas vraiment de travail préalable sur ce sujet. J'ai donc du écrire ce programme depuis le départ. Néanmoins les différentes méthodes d'accès étaient déjà identifiées, ce qui veut dire qu'il y avait quand même des connaissances préalables que m'a données mon maitre de stage.

I) Environnement de travail et théorie

1) Présentation du CERN

Le CERN est né de la volonté de pays européens de développer une véritable recherche nucléaire en Europe. Cette idée fut lancée à la fin de la 2ème guerre mondiale. CERN est en fait un acronyme pour Conseil Européen pour la Recherche Nucléaire. Ce conseil fut chargé d'étudier ce qu'il fallait développer comme infrastructure, et a proposé des lieux pour réaliser ce projet. Il fut formé en 1952 et dissout en 1954. Depuis, le CERN en tant que tel fut baptisé Organisation Européenne pour la Recherche Nucléaire, mais cette acronyme est resté attaché à cette organisation malgré la disparition de ce qu'il était censé représenter. **[1]**

Cette organisation est constituée de pays membres (Allemagne, Autriche, Belgique, Bulgarie, Danemark, Espagne, France, Finlande, Grèce, Hongrie, Italie, Norvège, Pays-Bas, Pologne, Portugal, République Tchèque, Royaume Unis, Slovaquie, Suède et la Suisse), de pays candidats (Chypre, Israël, Roumanie et Serbie), et de pays observateurs (Etats-Unis, Inde, Japon, Turquie et Russie). Il y a également de nombreux accords particuliers avec plus d'une quarantaine d'autre pays comme l'Australie, le Brésil, le Canada. **[2]**

L'objectif du CERN est de faire avancer la physique fondamentale en faisant des experiences. Dans ce but, une véritable structure a été aménagée sur deux sites : le principal, à Meyrin (Suisse) et le site de Prévessin (France). Plus de 11000 personnes travaillent quotidiennement sur le site de Meyrin. On pourrait parler de petite ville avec des réfectoires, tabacs, banques, agences de voyages, etc. Il n'y a

donc pas que des physiciens qui y travaillent, mais également des métiers annexes qui vont du cuisinier au pompier.

Pour arriver à ses fins, le CERN a développé plusieurs accélérateurs de particules. Le schéma suivant présente les différents accélérateurs encore en service.

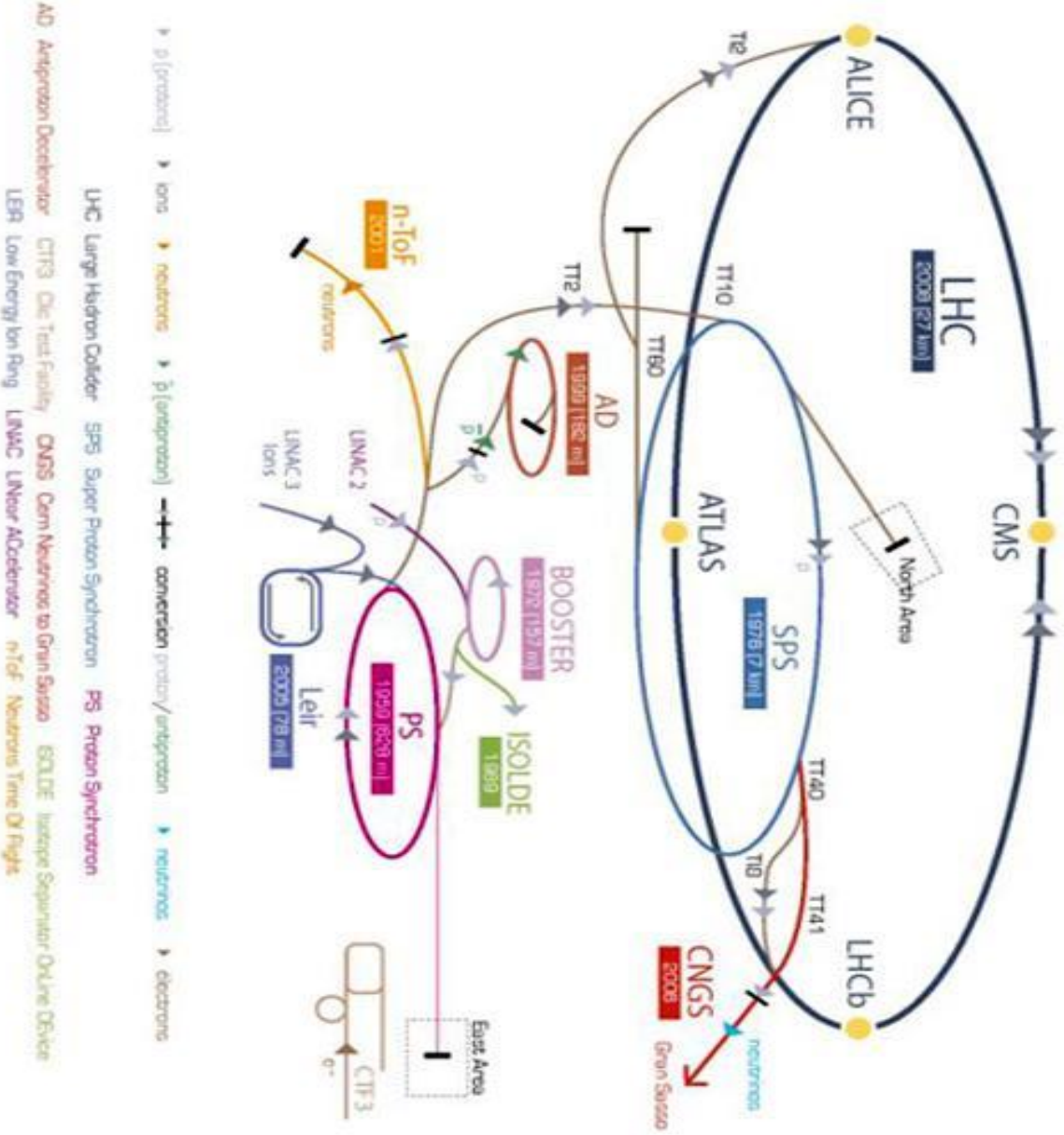

Figure 1 : schéma des différents accélérateurs de particules du Cern

Ils sont tous utilisés, certains pour d'autres expériences (comme le CNGS, Cern Neutrinos to Gran Sasso, qui envoie des neutrinos jusqu'aux environs de l'Aquila; c'est dans cette appareil qu'a été observée une particule qu'on a crû aller plus vite que la lumière), mais beaucoup comme premier accélérateur avant le LHC le plus grand et le plus récent de ces accélérateurs. Les particules passent par eux, afin d'acquérir une certaine vitesse/énergie avant d'arriver dans le LHC. Les particules sont en premier accélérées dans le LINAC 2 ou 3 (pour les ions de plomb) puis elles passent par le Booster ou le Leir (idem) et enfin il rejoigne le PS (Proton Synchrotron 1960) et puis le SPS (Super Proton Synchrotron 1971) avant d'être injecté dans le LHC. Le LHC, Large Hadron Collider, est le dernier accélérateur construit et est le plus puissant au monde. Il permet d'atteindre des énergies de l'ordre du TeV et des vitesses de 0,999999991 fois la vitesse de la lumière c'est-à-dire 2,7 mètres par seconde moins vite (sur les 299 792 458 m/s). Les faisceaux font à peu près 11000 fois le tour des 27km de circonférence du LHC par seconde. Le but est évidemment de provoquer des collisions dans les 7 détecteurs différents. Pour améliorer leur nombre et leur qualité le CERN a fait d'énorme progrès avec le LHC : le faisceau est désormais constitué de 2800 paquet avec un intervalle moyen de 25 nanosecondes entre eux. **[3]**

Le LHCb est l'un des quatre "grand" détecteur avec ALICE, CMS et ATLAS. LHC est comme mentionné plus haut, l'accélérateur de particules, et le b est pour beauté. Beauté du nom d'un des quarks, qui s'appelle effectivement quark beauty en anglais (le quark est le plus petit constituant de la matière découverte à ce jour). Et donc le LHCb a pour mission particulière de s'intéresser à cette particule et aux violations CP qu'elle engendre (c'est un terme scientifique qui identifie certaines différences entre matière et antimatière). **[4]**

Mon stage a été effectué au département Online, dirigé par Beat Jost. Ce département est responsable des données venant du détecteur LHCb. En fait ce secteur s'occupe du tri immédiat, il choisit les expériences potentiellement intéressantes et les enregistre sur des serveurs prévu à cet effet. La difficulté évidente est que les données arrivent en flux tendu et doivent être traitées en temps réel grâce à des algorithmes de tri extrêmement rapides (plus de 99.99% des résultats sont éliminés). Au final c'est tout de même 15 petaoctets de données qui sont sauvegardés par année pour l'ensemble du LHC. Un récent calcul effectué par mon tuteur, Loïc Brarda, donne un nombre de cœur de calcul de 15200 pour les serveurs LHCb online.

2) Le langage Python

Figure 2 : Logo du langage de programmation Python

Le Python est un langage de programmation de haut niveau, crée par Guido van Rossum, qui permet au programmeur de développer des logiciels de façon totalement indépendante des mécanismes de bas niveau. Cela était l'un des buts recherchés, pour que ce langage puisse être facilement appris, même par des

néophytes de la programmation. Ce langage fait partie des langages interprétés, c'est-à-dire qu'il n'y a pas de phase de compilation. En fait les modules non principaux sont compilés en bytecode, qui est un code intermédiaire. Il est doté d'un typage dynamique fort, et se sert d'un ramasse-miettes pour la gestion de la mémoire. Le système de ramasse-miettes est un système utilisé dans de nombreux langages comme le Java, le Ruby, le Perl, etc. Il consiste en fait à récupérer la mémoire qui n'est plus utilisée au fur et à mesure de l'exécution du programme. **[5]**

Le langage Python repose également sur un système de gestion des erreurs très efficace. En fait la philosophie prônée par Python est qu'il est préférable de gérer directement l'erreur. Par exemple, si une variable peut être vide et que l'on veut la manipuler, il est préférable de gérer cette erreur possible plutôt que de faire un test préalable pour voir si cette variable est bien vide.

Le Python est en majorité utilisé comme un langage de script. Il est apparemment massivement utilisé par la communauté des administrateurs système lorsqu'ils préfèrent éviter un script bash (Ruby et Perl sont également évoqués). Le python est néanmoins un langage multi-paradigme. C'est-à-dire qu'il permet d'écrire dans de nombreux styles de programmation comme l'impératif, orienté objet ou même fonctionnel.

3) La théorie sur les IPMI

L'IPMI, Intelligent Platform Management Interface, est un outil de gestion pour les serveurs réels. Cet outil fut développé par Intel, mais est devenu un standard utilisé

par une très grande majorité des constructeurs (Intel donc, HP, ASUS, Supermicro, etc.). C'est en fait une notion un petit peu confuse car cette acronyme désigne à la fois le service, le circuit réel et le protocole de communication interne.

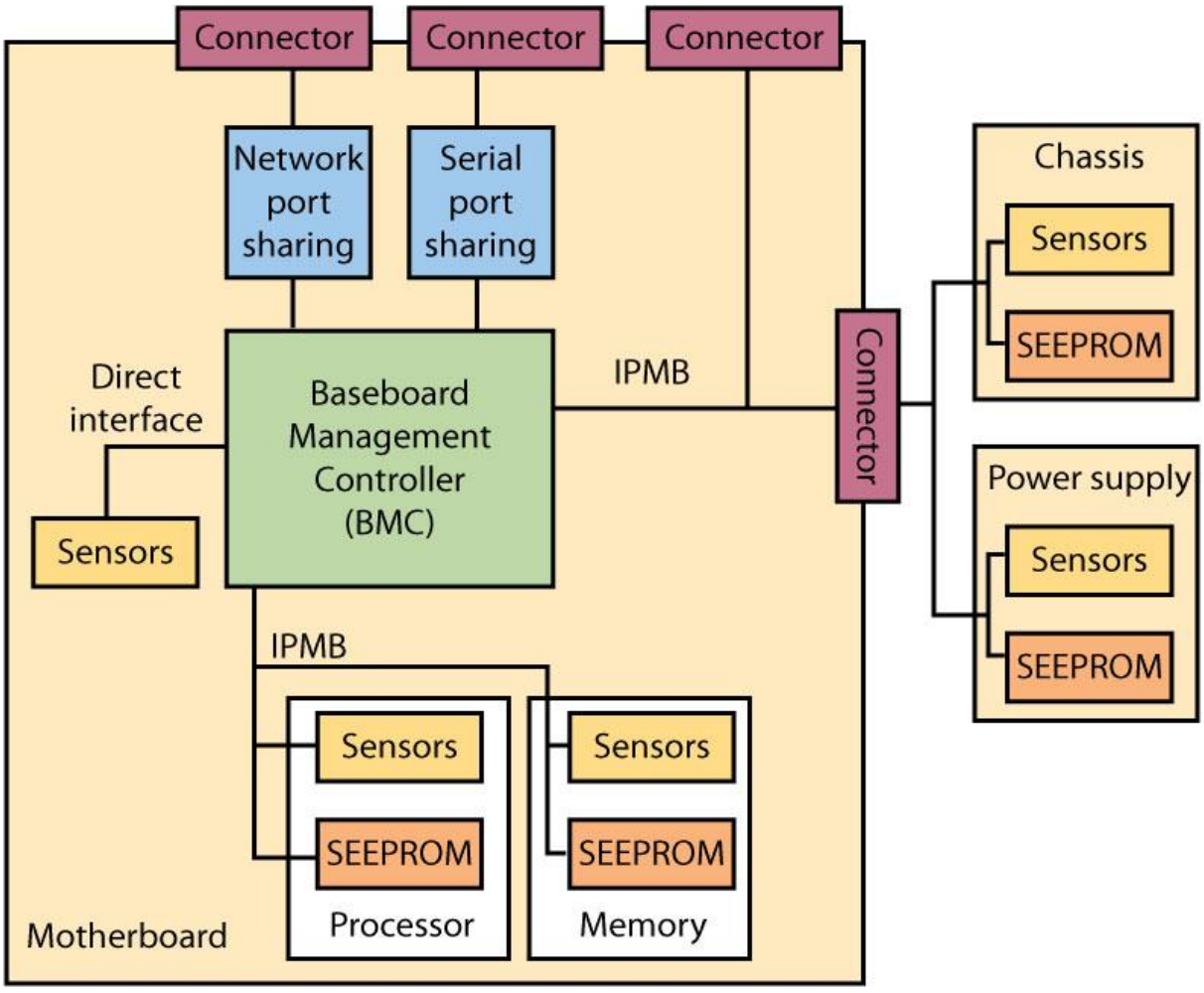

Figure 3 : Schéma de fonctionnement d'une IPMI

Ce schéma représente la généralité du principe de fonctionnement, car chaque constructeur met des petites particularités dans ce système. Néanmoins on peut voir

la pièce centrale appelée un BMC. Cet acronyme signifie Baseboard Management Controller. C'est en fait un microcontroller spécialisé qui est en général placé directement sur la carte mère du serveur. Il gère uniquement le système IPMI et travaille de façon indépendante du reste du serveur. On peut se connecter à cette BMC par deux moyens : soit en utilisant le réseau grâce à la carte réseau du serveur ou une carte qui lui est dédié, soit grâce à une liaison série directe. **[6]**

Ensuite il y a des capteurs disséminés dans tout le serveur. Il y a des capteurs de température pour les éléments critiques, la vitesse des différents ventilateurs, les niveaux de tensions électriques à différents endroits. Des alertes peuvent être déclenchées par des seuils limites sur les valeurs renvoyées au BMC. Ces capteurs sont reliés au BMC grâce à un bus très proche de la norme I2C qui s'appelle IPMB. Ce bus permet en fait de relier tous les systèmes de contrôle qui font partie de l'IPMI. **[7]**

Concrètement une IPMI est très utile lorsqu'il y a des problèmes graves sur le serveur. Par exemple s'il y a une erreur au boot (démarrage du serveur). Cela vient du fait que l'IPMI est complètement indépendante du système d'exploitation, et de tout le système de manière générale. C'est pour cela qu'a été développé l'IPMI ainsi que pour ses fonctions de surveillances des températures et des voltages.

Mais cela peut également devenir une faille de sécurité, car l'IPMI permet d'avoir accès, à distance, à un serveur quel que soit son état (même éteint). Il faut donc faire très attention aux politiques de sécurité qui doivent être utilisées. Dans beaucoup d'entreprises les IPMI sont connectées à un réseau indépendant du reste de l'entreprise. Cela permet de les cloisonner et de bien contrôler l'accès. Au CERN, ce n'est pas implémenté car les administrateurs en charge des IPMI préfèrent contrôler

les machines qui ont le droit d’accès à ces IPMI. En limitant le nombre de ces machines, cela devient assez aisé. Malheureusement, la technologie IPMI s’étant développée que récemment, il reste des failles de sécurité. **[7]**

Les constructeurs mettent à disposition de leurs clients une interface web assez simple d'usage, qui permet de juger de l'état du serveur, de configurer l'IPMI, de lancer une interface console en quelques clics.

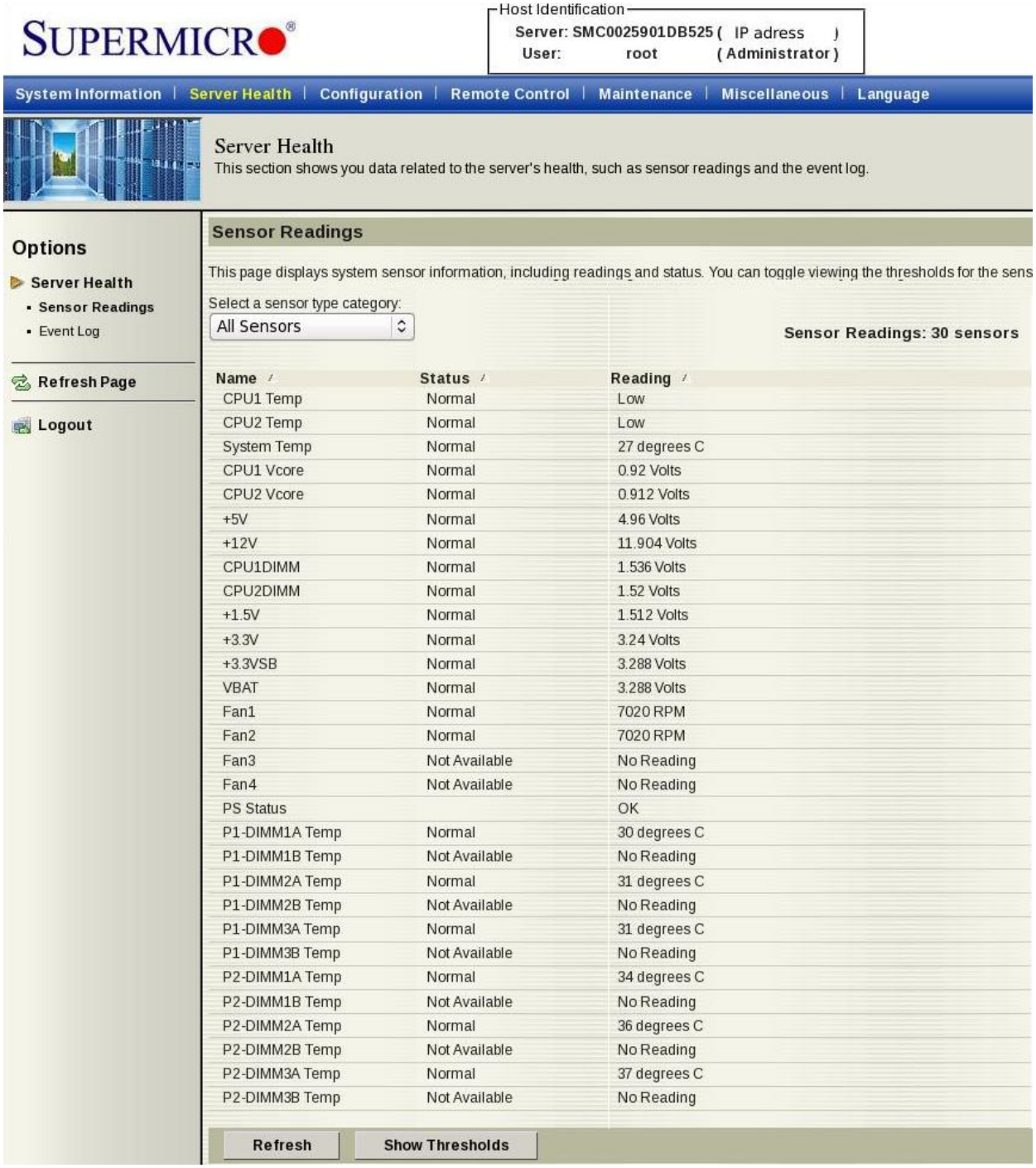

Figure 4 : Capture d'écran de l'interface web IPMI (modèle Supermicro)

Mais si une grande partie des serveurs ont désormais une IPMI, ils n'ont pas encore tous une interface web. Pour ceux-là un outil a été développé par Duncan Laurie qui s'appelle ipmitool. C'est un outil en ligne de commande très puissant qui permet d'envoyer tout un assortiment de commandes à l'IPMI.

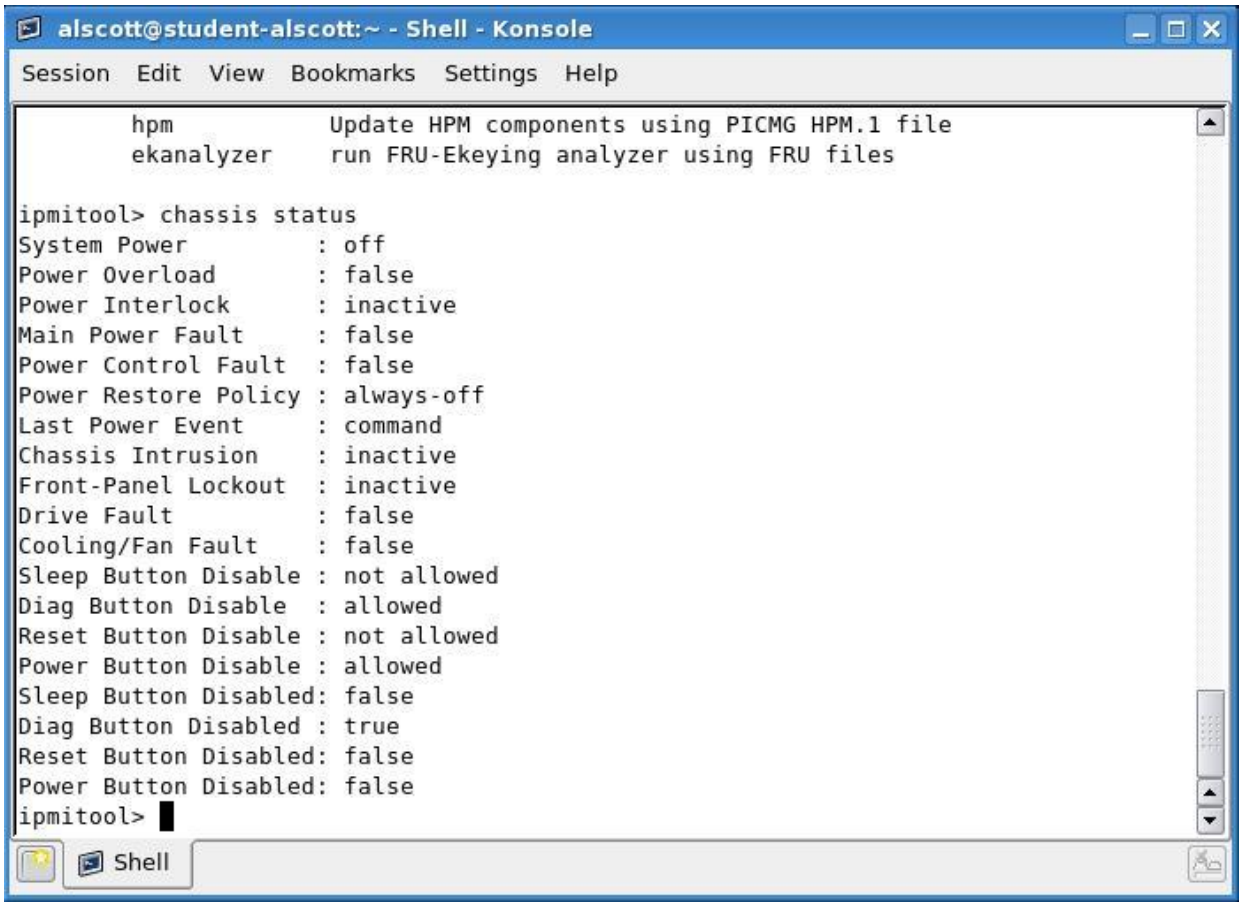

Figure 5 : Capture d'écran de l'interface ipmitool

On peut voir sur cette figure le shell ipmitool soumis à une requête sur le statut du châssis. On peut y voir que le système est éteint (System Power : off), ainsi que d'éventuelles problèmes d'alimentation, de refroidissement, et les différentes options

qui sont actives. Comme mentionné plus haut, cet outil est très complet et complexe. Il faudrait un rapport entier pour citer ses nombreuses fonctionnalités. Si cela vous intéresse, vous pouvez consulter la manpage d'ipmitool. **[8]**

Il y a donc ces deux méthodes d'accès, l'une dans un shell, et l'autre plus accessible car avec une interface graphique facile à manipuler.

4) La théorie sur la virtualisation

a . Les machines virtuelles

Une machine virtuelle est en fait un système d'exploitation qui est accueilli par un autre. Pour schématiser les choses on pourrait dire que ce système se trouve à l'intérieur d'une ''boîte'', contenue par un autre système d'exploitation.

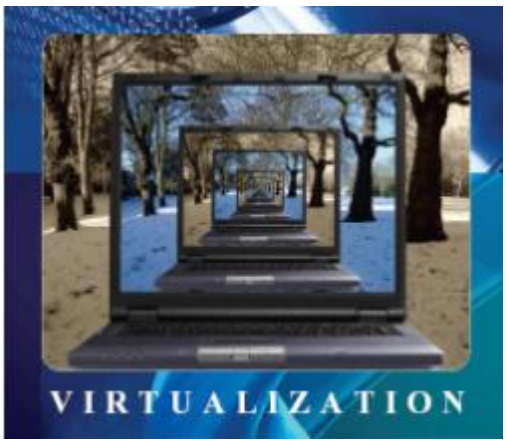

Figure 6 : Représentation photographique de la machine virtuelle

Néanmoins il faut bien qu'il y ait des méthodes pour que cette machine virtuelle puisse accéder aux ressources de la machine hôte (CPU, mémoire, réseau, etc). Cette boite n'est donc pas totalement hermétique, mais permet d'accéder aux ressources primordiales. Pour cela il faut une couche logicielle qui fasse croire à la machine virtuelle qu'elle a directement accès aux ressources, c'est une couche d'abstraction en fait.

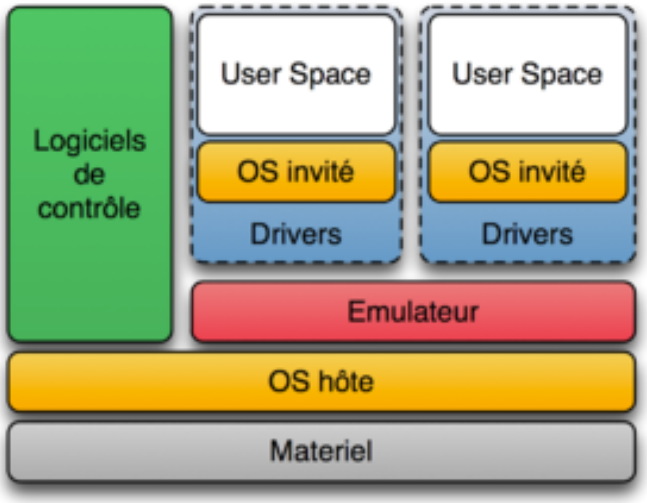

Figure 7 : Schéma de fonctionnement d'une machine virtuelle (ici il y en a deux)

Comme on peut le voir sur le schéma ci-dessus il y a un système d'exploitation (OS : Operating System) hôte qui se situe évidemment juste au-dessus du matériel, et ensuite il y a un ou des logiciels de contrôle qui distribuent les ressources. Sur ce schéma il y a un émulateur, mais celui-ci est souvent compris dans les logiciels de

contrôles, il sert à faire les appels systèmes (à l'hôte) lorsque les drivers des systèmes invités font des demandes matériel. Ce qu'il faut bien comprendre c'est que les systèmes invités n'ont pas conscience de l'être.

Il y a énormément d'utilité possible à ces machines virtuelles. Par exemple le langage Java utilise la JVM (Java Virtual Machine), et le C# utilise le système CLI (Common Language Infrastructure) pour se détacher du système d'exploitation hôte, et avoir une portabilité maximale grâce à cette abstraction matérielle. Ou alors si on veut utiliser des programmes douteux, par exemple des programmes en version bêta ou même avant cela et qu'il y a donc éventuellement des risques à l'utilisés sur une machine sensible, on peut utiliser une machine virtuelle. Si ce programme corrompt le système, il suffit de supprimer cette machine et d'en créer une autre.

L'avantage majeur pour le CERN est qu'une machine virtuelle ne peut pas avoir de souci de l'ordre des composants physiques, puisqu'elle n'en a pas... Si un serveur tombe en panne pour une raison ou pour une autre, il suffit de réaffecter le ou les serveurs virtuels sur un ou d'autres serveurs physiques. Bien sûr, cela est très important dans le cadre des expériences du CERN ou les données arrivent en continu et doivent être traitées immédiatement.

Un autre avantage très intéressant pour le CERN est que l'on peut mettre plusieurs de ces machines virtuelles sur une machine réelle. En effet une étude datant de 2008 a montré qu'en général 15% des capacités matérielles étaient utilisées. En sachant qu'un serveur ne consomme pas beaucoup plus d'énergie à 15% qu'à 90%, on voit bien qu'il est plus productif d'utiliser la virtualisation dont je vous parlerais plus en détail plus tard. Au CERN cet avantage est néanmoins moins net, car l'utilisation

des serveurs est très différente de ce qui pourrait se passer dans une entreprise classique.

b . La virtualisation

Actuellement la tendance à la virtualisation est très forte dans beaucoup de salles serveurs différentes. Ce qu'il faut bien comprendre avant de s'engager sur les techniques de virtualisation, c'est la raison d'être de cette avancée dans les technologies informatiques.

Tout d'abord qu'est-ce que la virtualisation : c'est en fait passer de serveurs réels à des serveurs virtuels, en installant les mêmes logiciels, et en leur affectant les mêmes travaux. Au départ ce n'est pas plus compliqué que de créer des machines virtuelles et de leurs assigner les tâches qui incombaient aux serveurs réels précédemment. Là où cette solution devient plus épineuse, c'est quand on a un grand nombre de serveurs virtuels à gérer.

Précédemment, il fallait un serveur pour chaque tâche, chaque sorte de service proposé. Certains de ces services occupent très peu des ressources serveurs, ou alors s'en servent mais à des moments relativement précis de la journée. Grâce à la virtualisation on peut rassembler des services sur un même serveur tout en gardant une frontière entre les types de serveurs. On peut donc économiser en matériel et énergies consommées tout en ne perdant pas de productivité (si la virtualisation est bien faite). **[9]**

De plus cela simplifie les systèmes de sauvegarde, les accélèrant et les rendant plus efficaces. Il y a d'ailleurs même une méthode où l'on crée une image d'un serveur pour créer une machine virtuelle que l'on n'utilisera que si ce serveur tombe en panne (cela s'appelle la méthode P2V Physical to Virtual). Cette méthode est apparemment très utilisée. **[10]**

Comme autres avantages, on peut noter le fait de centraliser la gestion des serveurs, la vitesse de déploiement ou de redéploiement de serveur (en cas d'erreur), minimiser les coûts en énergie et en refroidissement, et la possibilité de pouvoir tester des logiciels "suspects". Il y a également toutes les possibilités offertes par l'outil de gestion tel que la migration de machines lorsqu'elles tournent encore, ou le changement de caractéristiques comme la mémoire toujours lorsque les machines tournent (il faut néanmoins faire très attention), et de manière bien plus aisé que sur un serveur physique. **[9]**

Mais il ne faut pas non plus croire que la virtualisation est une solution miracle. Il y a des désavantages. Imaginons par exemple que l'on veuille conserver des machines virtuelles sur un serveur en particulier. Si celui-ci a des problèmes physiques graves, toutes les machines virtuelles sont misent hors service. Il faut également comprendre que la virtualisation demande des ressources et est donc inefficace sur des serveurs bas de gamme. Et enfin certaines applications peuvent connaître des soucis de performance. Par exemple les serveurs de bases de données sont connus pour perdre en performance lorsqu'ils sont installés dans un système virtualisé. Car une des limitations les plus restrictives sont les entrées/sorties du serveur physique. **[9]**

De plus la virtualisation demande un système de stockage bien dimensionné. Car en multipliant les machines virtuelles, on multiplie également les demandes pour des ressources présentes dans la mémoire morte.

Pour gérer ce véritable parc de machines virtuelles, il faut donc utiliser des outils adaptés, comme un gestionnaire pour faciliter toutes les opérations d'installation et de maintenance. Il en existe quelques-uns comme les systèmes produits par VMWare (le leader du marché), l'Hyper-V de Microsoft, ou Xen. Le CERN a choisi le système produit par Red Hat, RHEV. Red Hat Enterprise Virtualization possède des outils permettant de faire une gestion simple et efficace de l'ensemble de ces machines; iI met à disposition une interface web que vous pouvez voir sur le schéma suivant.

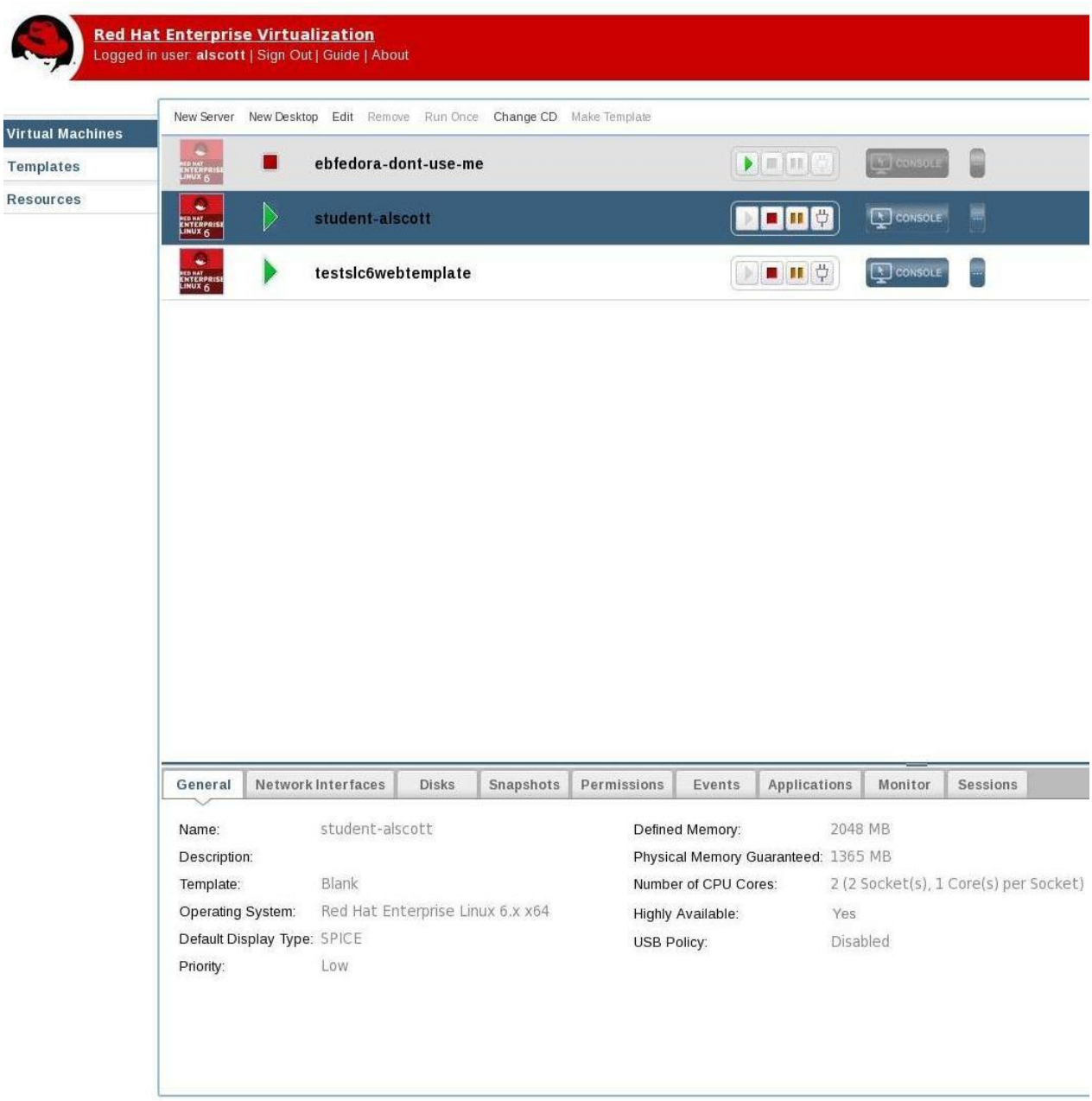

Figure 8 : Capture d'écran de l'interface web de RHEV (coupée)

Sur cette capture d'écran il n'y a que 3 machines virtuelles car suite à une série de mises à jour, je n'avais plus les droits sur les autres. Il y en a évidemment bien plus,

de l’ordre de 300. Et ce nombre risque d’augmenter car la virtualisation de la salle serveur se fait petit à petit.

Par ailleurs les fonctionnalités de RHEV sont également disponibles grâce à un petit utilitaire appelé ovirt-shell, Ce logiciel fonctionne comme un shell, qui permet de faire ce dont on a besoin en ligne de commande. Il est à noter que cet outil est issu de oVirt qui est la version “upstream” de RHEV, c'est-à-dire que les innovations sont d'abord testées dans oVirt avant de passer dans RHEV, une fois qu'elles sont jugées stables (comme Fedora et RedHatEntrepriseLinux).

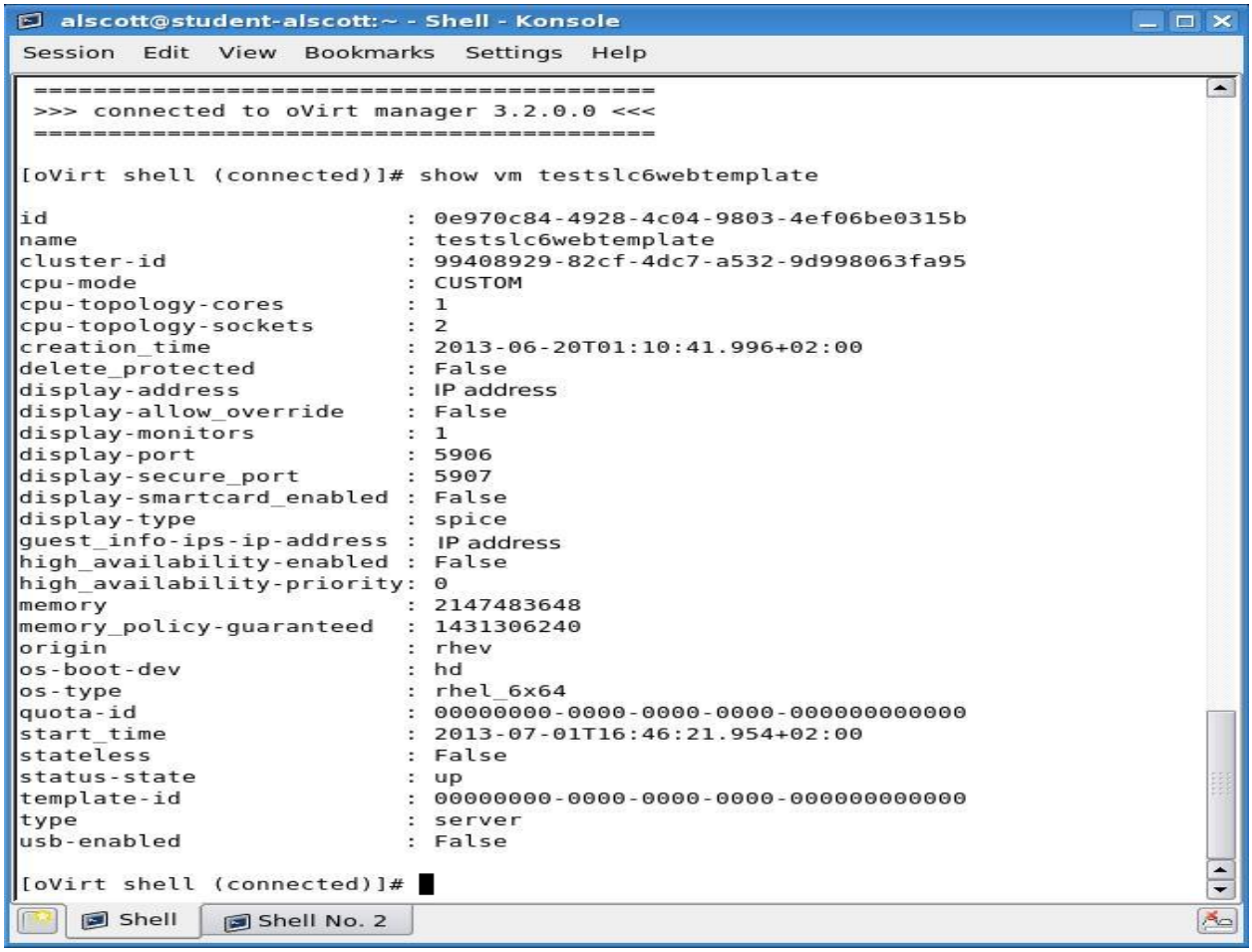

Figure 9 : Capture d'écran lors de l'utilisation d'ipmitool

Voilà les deux accès différents à l'utilitaire de gestion des machines virtuelles.

De plus ces outils permettent de faire beaucoup de choses de manière très simple. Ainsi on a accès à des fonctionnalité telles que créer une machine virtuelle à partir d'un ‘’template’’ (une machine virtuelle qui sert de modèle), déplacer une machine virtuelle, ou changer la quantité de ressources allouées à une machine virtuelle. Le

point le plus intéressant qui a été rajouté à la version 3.1 de RHEV est la “live migration”. Cela permet de déplacer une machine virtuelle d’un serveur à un autre sans devoir l'arrêter, et sans arrêter du même coup les services qu’elle propose. C’est notamment très intéressant en cas de problème avec un serveur physique. Il existe également une option qui permet de faire de l’ “automatic balancing”, c’est-à-dire de laisser RHEV gérer le parc en déplaçant des machines virtuelles pour essayer d'homogénéiser l’utilisation des serveurs (cela peut néanmoins être dangereux car on ne sait désormais plus où se situent certaines machines, et le balancing peut être fait à un moment où l’utilisation est faible et donc rassembler sur un même serveur plusieurs machines très utilisées en temps normal). **[11]**

Ce système s'appuie sur ce qui s’appelle un hyperviseur (de l’anglais hypervisor) qui s’appelle kvm (Kernel-based Virtual Machine). Un hyperviseur est une sorte de plate-forme de virtualisation. Il sert à faire suivre les requêtes matérielles des systèmes invités en les optimisant. Il existe en fait deux types d’hyperviseur, le type 1 (aussi appelé natif ou bare-metal) et le type 2. **[12]**

Le type 1 est directement installé sur le matériel. Il est accessible uniquement par une machine virtuelle installée d’office qui est nommée domain0.

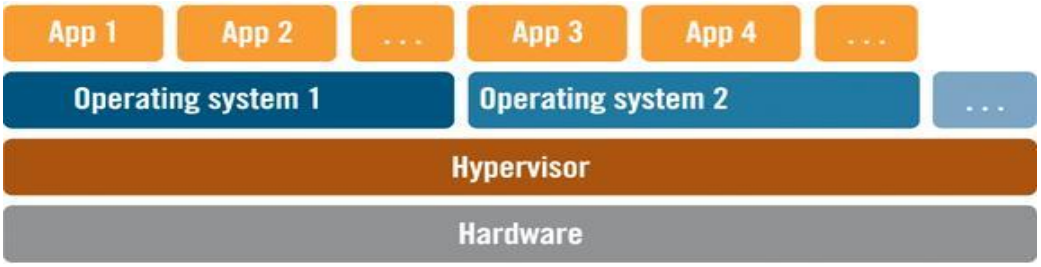

Figure 10 : Schéma de fonctionnement de la virtualisation avec un hyperviseur de type 1

Un hyperviseur de type 2, lui, est situé au-dessus d'un système d'exploitation hôte.

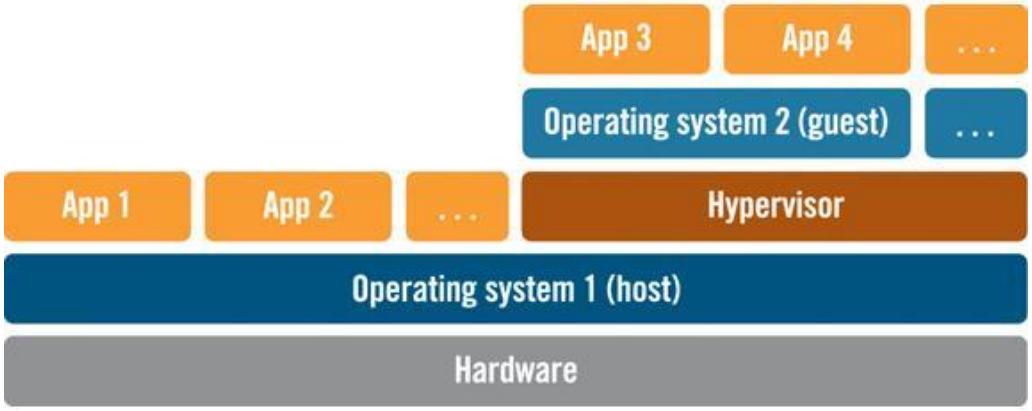

Figure 11 : Schéma de fonctionnement de la virtualisation avec un hyperviseur de type 2

La plate-forme KVM est considérée comme un hyperviseur de type 1, car bien qu'étant intégrée à un Linux, elle fait partie du noyau de celui-ci et transforme de ce fait le système d'exploitation en un hyperviseur.

II) Conception de la solution

Le but de mon stage était d'effectuer une connexion à des serveurs de manière directe, c'est-à-dire sans que l'utilisateur n'ait besoin de savoir de quel type de serveur il s'agit. Il en existe 2 grands types, les réels et les virtuels. J'ai donc créé deux modules, un pour chacun. Et comme déjà indiqué, les IPMI n'ont pas toute une redirection console (par l'interface web), il faut donc utiliser une autre méthode (le SOL). De ce fait mon module IPMI se divise en deux fonctions, une pour les IPMI ayant une redirection console et une autre pour ceux qui ne l'ont pas. Ceci amène à l'ossature de programme décrite dans le schéma suivant. Mais il reste à trouver une méthode pour savoir quel serveur est de quel type.

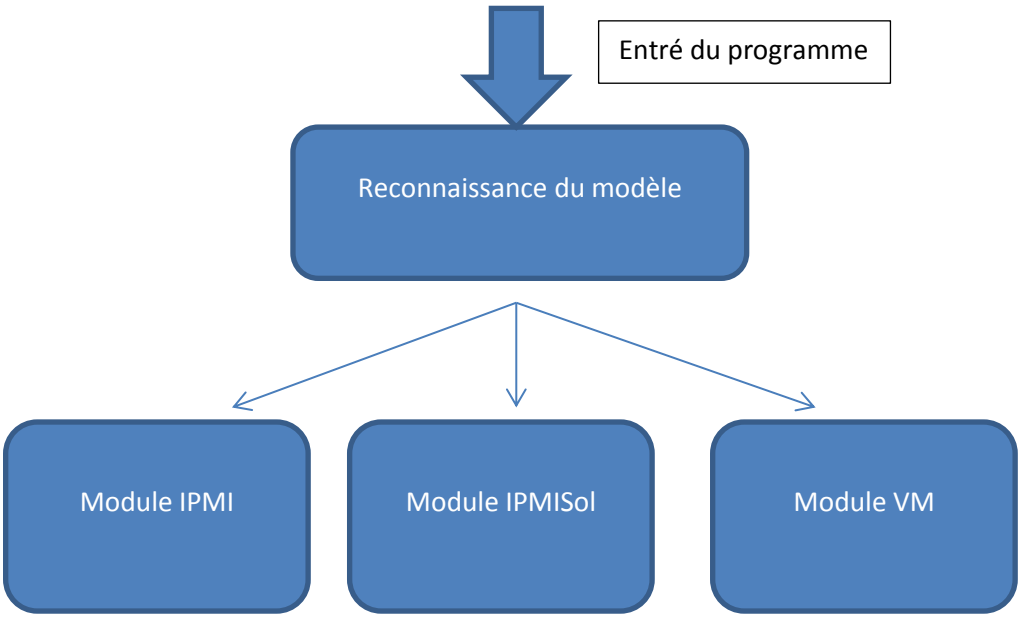

Figure 12 : Schéma de principe de mon programme

Pour cela il a fallu déterminer pour chaque modèle quelle étais la méthode d'accès adéquate. Le programme dois donc garder en mémoire quelle méthode est utilisé pour quel modèle. L'association entre modèles et serveurs vient d'une base de données. On commence donc à comprendre le besoin de créer une interface avec cette base de données, pour récupérer le modèle et ensuite lancer la bonne fonction.

1) Les IPMI

Pour le module IPMI, il y a deux fonctions séparées. Tout d'abord j'expliquerai la partie sur les IPMI avec une interface web. A cet effet, les outils web du langage Python ont été utilisés pour se connecter à l'interface web.

La stratégie était d'imiter ce que faisaient les interfaces web des constructeurs. A cette fin, la Web Console incorporée dans le navigateur Firefox **[13]**, ainsi que le logiciel connu des personnes faisant du réseau, Wireshark **[14]**, ce sont avérés très utiles. Malheureusement beaucoup de ces sites exigeaient une connexion avec le protocole https (la version sécurisée de http), et donc rien ne pouvait être vu avec Wireshark.

Il a fallu en fait créer une session sur ces sites, en envoyant les bonnes informations (comme le nom d'utilisateurs et le mot de passe par exemple) par requête POST. Le problème, au départ, était que je n'avais pas de documentation portant sur les noms des variables à utiliser. Comme déjà mentionné, Wireshark était inutilisable dans certain cas, et il n'y avait pas d'envois sous forme de cookies. Pour finir, j'ai réussi à trouver ces noms en cherchant dans le code source des fichiers JavaScript. Il a fallu

répéter trois fois cette opération car les interfaces diffèrent d'un constructeur à l'autre.

Ensuite le service renvoie un identifiant de session dont il faut se servir pour toutes les demandes de pages par la suite. La difficulté à ce point est que la méthode d'envoi n'est pas la même partout. Certain constructeur utilise un cookie (qu'il faut donc récupérer) et d'autres renvoient une page sous un format JSON. Le JSON (JavaScript Object Notation) est une méthode pour stocker et récupérer des données sous forme de paires nom, valeur. Le délimiteur en est les accolades ({, }). **[15]**

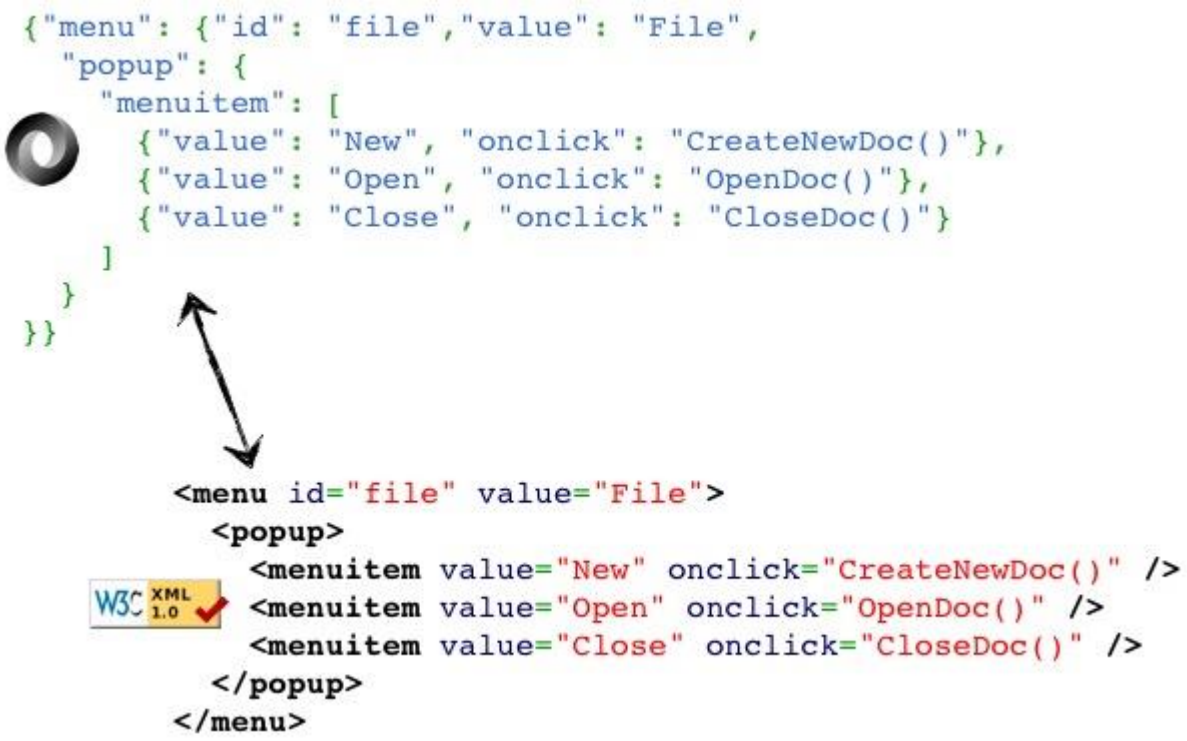

Figure 13 : Comparaison entre deux formats de sérialisations

Sur ce schéma, vous pouvez voir en haut un exemple de JSON et en bas le même exemple mais pour le format xml.

Normalement, la sérialisation et désérialisation du JSON est très simple et est censée être indépendante du langage (bien que venant de JavaScript comme l'acronyme nous l'indique, plus de 55 langages le supportent). Malheureusement, dans le cas présent, ce ne fut pas si simple. Il a fallu des manipulations préalables avant de lancer la fonction de désérialisation. Par exemple, le JSON demande des guillemets doubles (“), tandis que la page récupérée du constructeur n'avait que des guillemets simples (‘).

Une fois ce numéro de session obtenu, il faut faire une demande pour récupérer la page contenant le JNLP (Java Network Laucher Protocol) qui doit être ensuite enregistrée dans un fichier à l'extension .jnlp. Ce fichier est un outil de la technologie Java Web Start, qui permet de lancer une application Java depuis un client (qui à Java installé). Ce fichier est en fait un simple fichier XML, où sont référencés des éléments nécessaires à l'application, comme où sont situées les librairies jars, les arguments, etc. **[16]**

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp codebase="http://fudaqs2.ciemat.es/TJ2WEB/">
    <information>
        <title>SimilWave App</title>
        <vendor>TJ2WEB Data Acquisition Group</vendor>
    </information>
    <security>
        <all-permissions/>
    </security>
    <resources>
        <j2se version="1.5+" />
        <jar href="bin/similarwave.jar" />
        <jar main="true" href="bin/papiCookieManager.jar" />
        <property name="Resources" value="http://fudaqs2.ciemat.es/TJ2WEB/similarwave" />
    </resources>
    <application-desc main-class="papiUtils.Runner">
        <argument>similarwave.SimilarWave</argument>
        <argument>http://fudaqs2.ciemat.es/TJ2WEB/similarwave/</argument>
    </application-desc>
</jnlp>
```

Figure 14 : Exemple de fichier JNLP

Comme on peut le voir, il y a plusieurs sections qui se retrouvent toujours dans un JNLP: “information”, “security”, “ressources”, “application-desc”. La borne “information” donne des informations relatives au produit, “security” définit le niveau de sécurité, “resources” donne le chemin d'accès aux ressources, et “applications-desc” regroupe en général les paramètres donnés au programme. **[16]**

Par la suite il n’y a plus qu’à lancer ce fichier à l’aide de l’utilitaire Java qui s’appelle javaws (Java Web Start). Cela lancera la console de redirection, qui est codée par le constructeur en Java (vous vous en doutiez j’imagine). Il a fallu par contre, trouver la bonne version de Java pour que cela marche, car c’est assez erratique. Chaque interface est en général compatible avec quelques versions de Java, mais c’est tout.

Finalement, je me suis rendu compte que la version 1.6.0_18 convenait à toutes les interfaces.

2) Les IPMI sans interface web

Pour les serveurs qui ont une IPMI mais n'ont pas d'interface web, je suis passé par l'outil ipmitool. Le faite de lancer une session SOL est assez simple ici, puisqu'il n'y a qu'à lancer ipmitool avec les bons arguments :

ipmitool -I lanplus -H SERVER(-ipmi) -U LOGIN -P PASSWORD sol activate

Comme déjà mentionné, les IPMI ont un nom (hostname), qui est au CERN composé du nom du serveur sur lequel il se situe avec -ipmi en plus.

Il faut souligner qu'après avoir activé une session SOL (''activate'' dans la commande ci-dessus) il faut la désactiver, ce qui ne se fait absolument pas automatiquement. Cette désactivation est obligatoire car il ne peut y avoir qu'une connexion SOL par serveur, donc si on ne la désactive pas après avoir fini son travail personne ne pourra se connecter par la suite. Pour cela il suffit de reprendre l'instruction écrite plus haut en remplaçant ''activate'' par ''deactivate''.

Par contre, à la demande de mon tuteur, il a fallu gérer globalement les sessions SOL. C'est-à-dire savoir qui a une session ouverte, depuis quelle machine, et vers quelle machine. Cette fonction a été implémentée via un fichier texte classique qui stocke ces informations à chaque connexion et les retire à chaque déconnexion.

```
This file contains all the SOL connection that are issued through the unifiedAccess program
########time########;##hostname(from)##;##hostname(to)##;##pid##;###user###

23/07/2013 09:38:57 ; student-alscott ; hlta0606-ipmi ; 23085 ; alscott ;
23/07/2013 09:39:10 ; student-alscott ; hlta0607-ipmi ; 23092 ; alscott ;
23/07/2013 09:39:24 ; student-alscott ; hlta0608-ipmi ; 23099 ; alscott ;
23/07/2013 09:39:38 ; student-alscott ; hlta0609-ipmi ; 23106 ; alscott ;
```

Figure 15 : Capture d'écran du contenu du fichier IPMITrack

Sur cette capture d'écran, on voit le contenu d'IPMITrack, qui est le fichier utilisé pour stocker ces informations. On voit également quatre sessions ouvertes sur quatre serveurs différents dont je suis l'utilisateur. Le PID de cette session est également enregistré, car cela me permet d'introduire une autre fonctionnalité : la possibilité de terminer la session qui peut bloquer l'accès à l'IPMI d'un serveur. Pour cela il a fallu se servir d'une méthode d'accès à distance. J'ai choisi le SSH car c'est le plus simple d'accès et le plus facile à utiliser au final.

sudo ssh -l LOGIN HOSTNAME kill PID

Avec LOGIN le nom d'utilisateur de la personne qui veut tuer la liaison SOL, HOSTNAME la machine sur laquelle a été lancé le processus et PID le numéro d'identification de ce processus de liaison SOL. Ceci est possible car ce programme s'adresse à des administrateurs qui ont les droits nécessaires pour pouvoir terminer un processus qui n'est pas le leur. C'est-à-dire que l'instruction sudo leur octroiera les droits quelle que soit la machine. Pour ceux qui ne disposent pas de ces droits et veulent utiliser mon programme, je ne pense pas que ce soit une bonne idée de leur donner la possibilité de terminer la session d'un autre utilisateur.

3) Les machines virtuelles

Pour les machines virtuelles, l'outil privilégié par le CERN s'appelle spicec, et il lance une console SPICE. Mais pour cela il y a besoin de récupérer les informations nécessaire sur la machine que détient le superviseur RHEV (l'addresse IP, les numéros de ports, un ticket d'authentification, etc.). Après beaucoup de recherches, nous avons trouvé une API python pour ovirt, la version upstream de RHEV. Grâce à celle-ci on peut relativement facilement trouver ces informations sur la machine que l'on veut contacter. Le problème qui est apparu est que cette API fait partie des versions tests de RHEV, et donc n'est pas exempt de bugs.

Ce qui a demandé plus de travail est la gestion des certificats pour la connexion au maître du serveur. En effet au départ l'accès était systématiquement refusé pour d'autres maîtres que celui qui a été initialement testé. La façon de faire est de gérer les certificats des maitres grâce au fichier spice_truststore.pem qui se situe dans le répertoire caché .spicec (le point signifie que c'est un fichier caché pour beaucoup de système Unix) sur le répertoire ''maison'' de l'utilisateur. Il faut en fait le mettre à jour à chaque fois que l'utilisateur se connecte à un nouveau maitre.

4) Les autres modules nécessaires

Au-delà de la structure déjà décrite, il y a besoin de créer des modules annexes qui servent les modules détaillés plus haut. Ceci pour éviter les répétitions de codes et la surcharge de ces modules.

Le module connectDB gère toutes les connexions à la base de données. En fait il n'y a besoin que d'accéder à une table, celle qui répertorie l'ensemble des serveurs du LHCB online avec comme attributs le nom du serveur, le modèle de ce serveur, l'hyperviseur auquel il est rattaché que l'on appelle maître (pour les serveurs virtuels), le ou les groupes auquel appartient ce serveur et le ou les mots clefs qui lui sont associés. Il y a bien d'autres champs mais je ne les détaille pas car il ne nous intéresse pas dans le cadre de mon stage.

Je vais maintenant détailler pourquoi ces attributs sont importants dans notre cas. Comme déjà soutenu, le modèle du serveur est primordial pour savoir quelle sorte de console il faut lancer par la suite (IPMI par l'interface web, IPMI sans interface web par ipmitool et les machines virtuelles par spice). Le maître lui, est important pour les machines virtuelles, il faut savoir sur quelle interface se connecter. Le groupe et le mot-clef ne sont pas vraiment primordiaux pour mon programme, mais ils peuvent aider les utilisateurs.

Le module configParser sert à gérer les paramètres passés en ligne de commande au programme, en utilisant le module argparse de Python. Pour utiliser le programme il faut fournir un nom de serveur et un autre argument que j'appellerai ‘’demande’’. Cette demande doit être l'un des modes suivants : console, status, detail, suspend, resume, start, shutdown, stop, groups, keywords, model, master. ‘’Console’’ est le

mode principal qui a été détaillé précédemment, ‘’status’’ et ‘’detail’’ sont des raccourcis, et ‘’groups’’-‘’master’’ ce sont des outils de requête à la base de données.

Les ‘’demandes’’ qui sont appelées ici des raccourcis sont des requêtes que l’utilisateur peut faire facilement s’il en a envie grâce à la console. En fait le programme fait directement la requête pour l’utilisateur, ce qui se révèle utile puisque ce sont des instructions particulièrement utilisées. ‘’Status’’ et ‘’detail’’ sont des requêtes pour obtenir des informations sur un serveur, ‘’status’’ donne juste l'état de la machine (en marche, éteint ou suspendu), et ‘’detail’’ donne un peu plus d’information. L’effet des instructions de ‘’suspend’’ à ‘’stop’’ sur les machines est plutôt évident. La différence entre ‘’stop’’ et ‘’shutdown’’ mérite toutefois d'être expliqué : ‘’stop’’ initie l'arrêt immédiat du système quoiqu’il se passe, alors que ‘’shutdown’’ éteint proprement le système en attendant qu’il soit prêt à être éteint.

Les autres instructions se servent des fonctions déjà écrites dans le module connectionDB. La seule subtilité est qu’elles renvoient le(s) groupe(s), mot-clef(s), modèle, maître d’un serveur s’il est fourni en ligne de commande, ou la liste de tous les éléments de cette nature (groupe, etc.) si aucun serveur n’est fourni.

Le module connectionUtilities me sert pour tout ce qui est de l’ordre des connexions aux interfaces web. C’est-à-dire qu’il y a des fonctions pour envoyer des requêtes POST, certaines pour envoyer des requêtes GET, et d’autres pour gérer les cookies. Cela est nécessaire pour une connexion à l’IPMI par l’interface web.

Pour la gestion des fichiers en général, j’ai créé un module fileUtilities. Ce module gère la création et la destruction de fichier, la desérialisation du JSON.

Le module myExceptions sert à gérer les cas d'erreurs, notamment en formatant le message initial pour inclure le module d'origine avec la fonction et la ligne.

Par la suite j'ai également détaché du module principal les appels aux modules de traitement pour faciliter la maintenance de mon programme. Il a donc fallu créer un module modelList comprenant la liste des modèles supportant IPMI avec interface web, celle supportant IPMI sans interface et celle des machines virtuelles.

III) Résultats et discussions

Le programme réalisé répond à la consigne. C'est-à-dire qu'il permet de joindre un serveur qu'il soit réel ou virtuel. Mais il reste quelque bugs éparses que je vais décrire plus loin. De plus il a fallu réaliser une interface graphique.

1) Interface graphique

L'intitulé du stage comportait la réalisation d'une interface graphique minimaliste pour ce programme. Je me suis donc d'abord posé la question de comment la réaliser, et quels outils utiliser. Évidemment ces outils doivent être compatibles avec le langage Python puisque c'est le langage que j'ai utilisé pour écrire le reste du programme. De cette constatation trois possibilités ce sont dégagées, un outil Python, Tkinter, et deux outils multi-langages GTK+ et QT.

Tout d'abord j'avais déjà fait du QT (pour un projet en C++), et puisque j'étais en avance sur mon planning, j'ai décidé qu'il serait plus utile de découvrir d'autres méthodes.

Tkinter est un outil relativement ancien. Après en avoir fait un peu je me suis rendu compte que cela ne répondait pas vraiment à mes besoins. J'ai donc laissé cet outil de côté.

Je me suis donc rapidement penché sur GTK+ (la traduction en Python étant assurée par le module PyGtk) qui est un outil graphique puissant utilisé notamment

pour la plateforme GNOME. J'ai également utilisé un logiciel qui s'appelle glade (outil RAD : Rapid Application Development), qui permet de faire une interface graphique relativement simplement en positionnant les éléments voulus dans la fenêtre (similaire à Microsoft Visual Studio sur ce point). Le gros avantage de ce logiciel est qu'il produit un fichier .glade, qui est en fait un fichier XML, contenant toute l'interface graphique. Ensuite PyGtk fournit un outil de désérialisation, ce qui simplifie énormément le code de l'application, car il n'y a plus qu'à coder ce qu'il y a derrière le graphisme (lien avec la base de données par exemple) et récupérer les signaux des éléments graphiques déclarés dans le .glade. **[17]**

Cette interface se veut simple d'utilisation et apporte des petits ajouts par rapport à la version par ligne de commande qui augmente la magnabilité du logiciel.

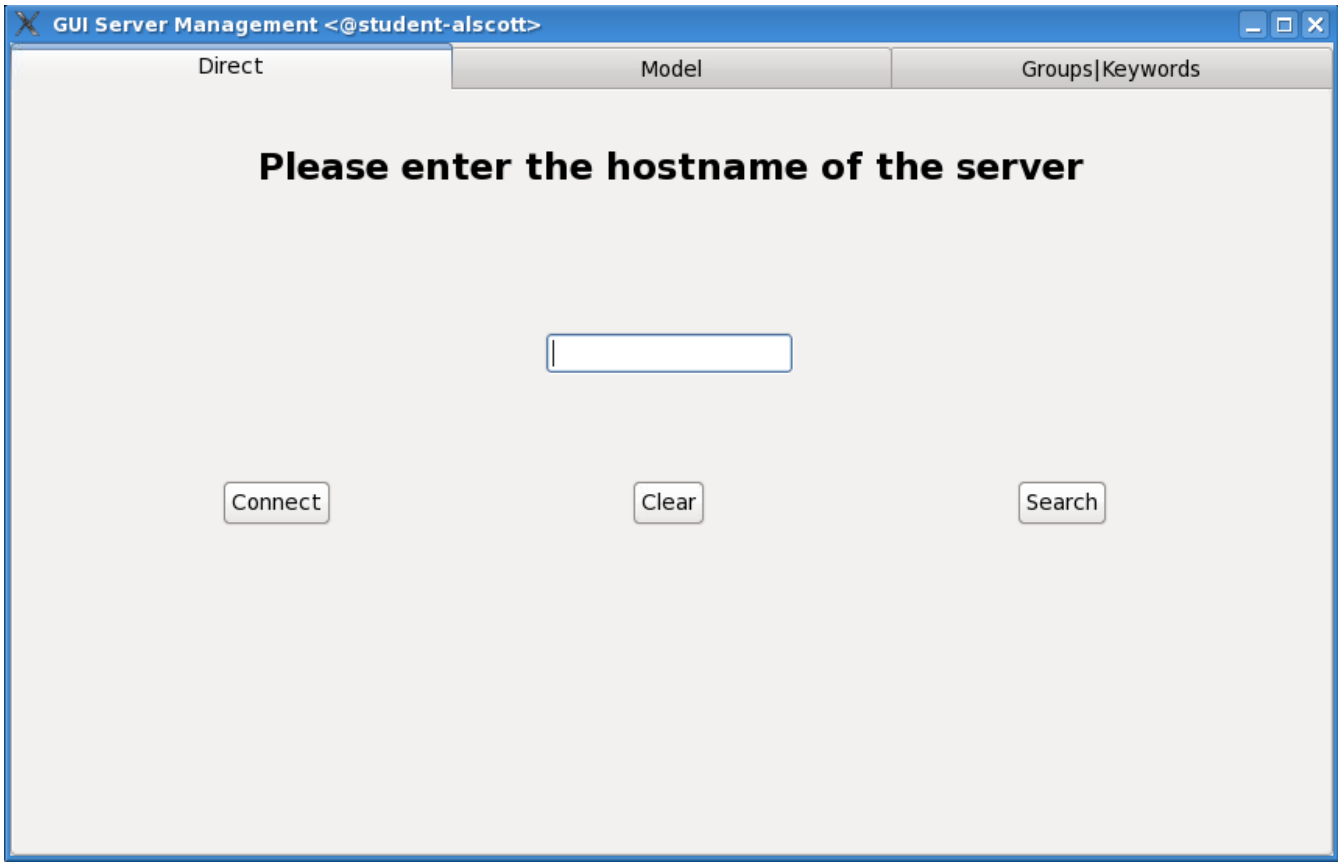

Figure 16 : Capture d'écran de l'application graphique (accueil)

On peut voir ici l'application graphique. Elle a un design assez sommaire. Elle se décompose en trois onglets, ''direct'', ''model'', et ''groups keywords''. Sur ce premier onglet vous pouvez voir un champ permettant de rentrer le nom d'un serveur. J'ai mis en place un système d'auto-complétion :

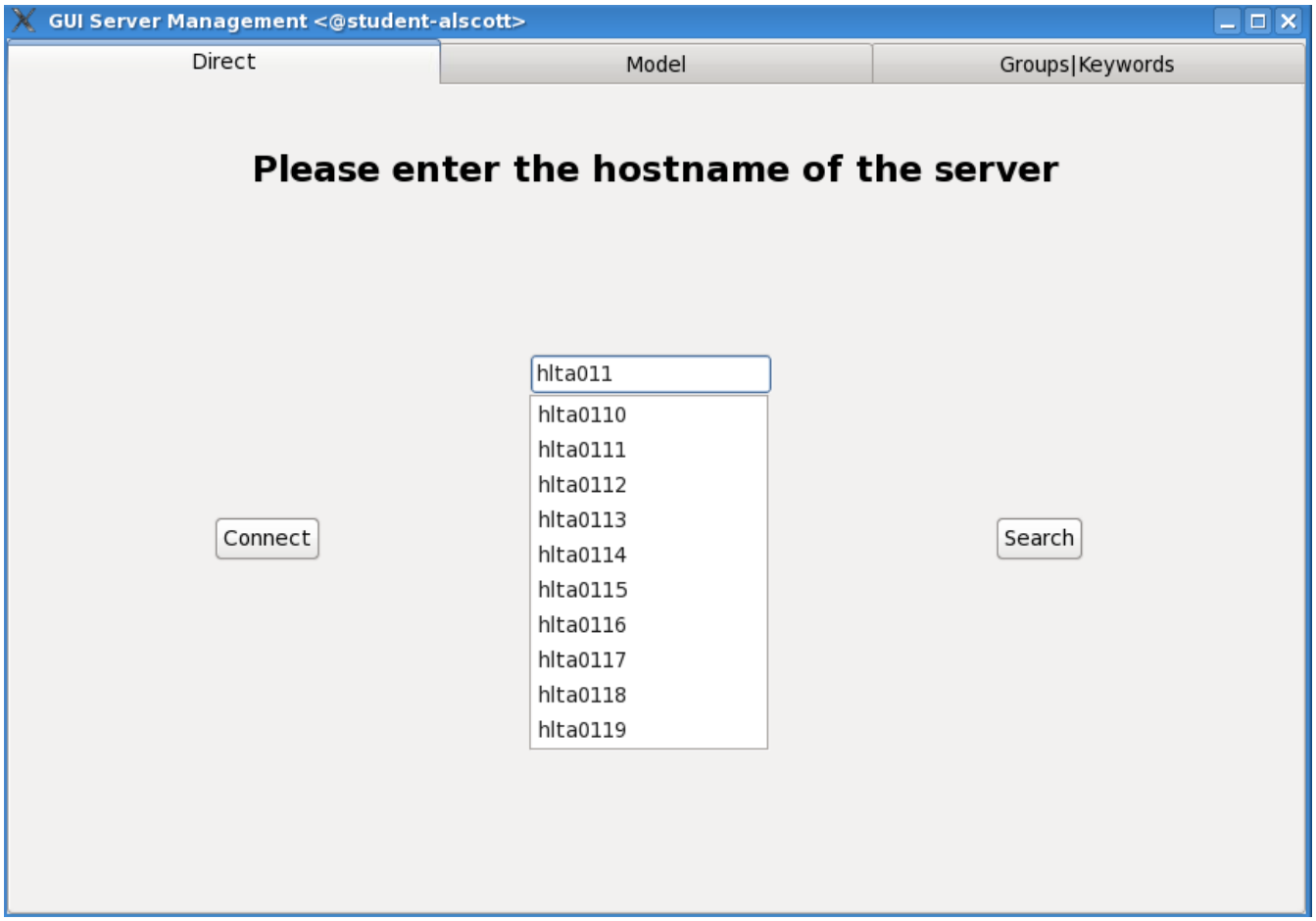

Figure 17 : Capture d'écran de l'application graphique (auto-complétion)

On peut également voir des boutons. Le bouton ''connect'' sert évidemment à créer une console vers le serveur sélectionné, ''clear'' (invisible sur la dernière figure) sert à nettoyer le champ, et ''search'' fait apparaître les quatre informations potentiellement importantes sur le serveur :

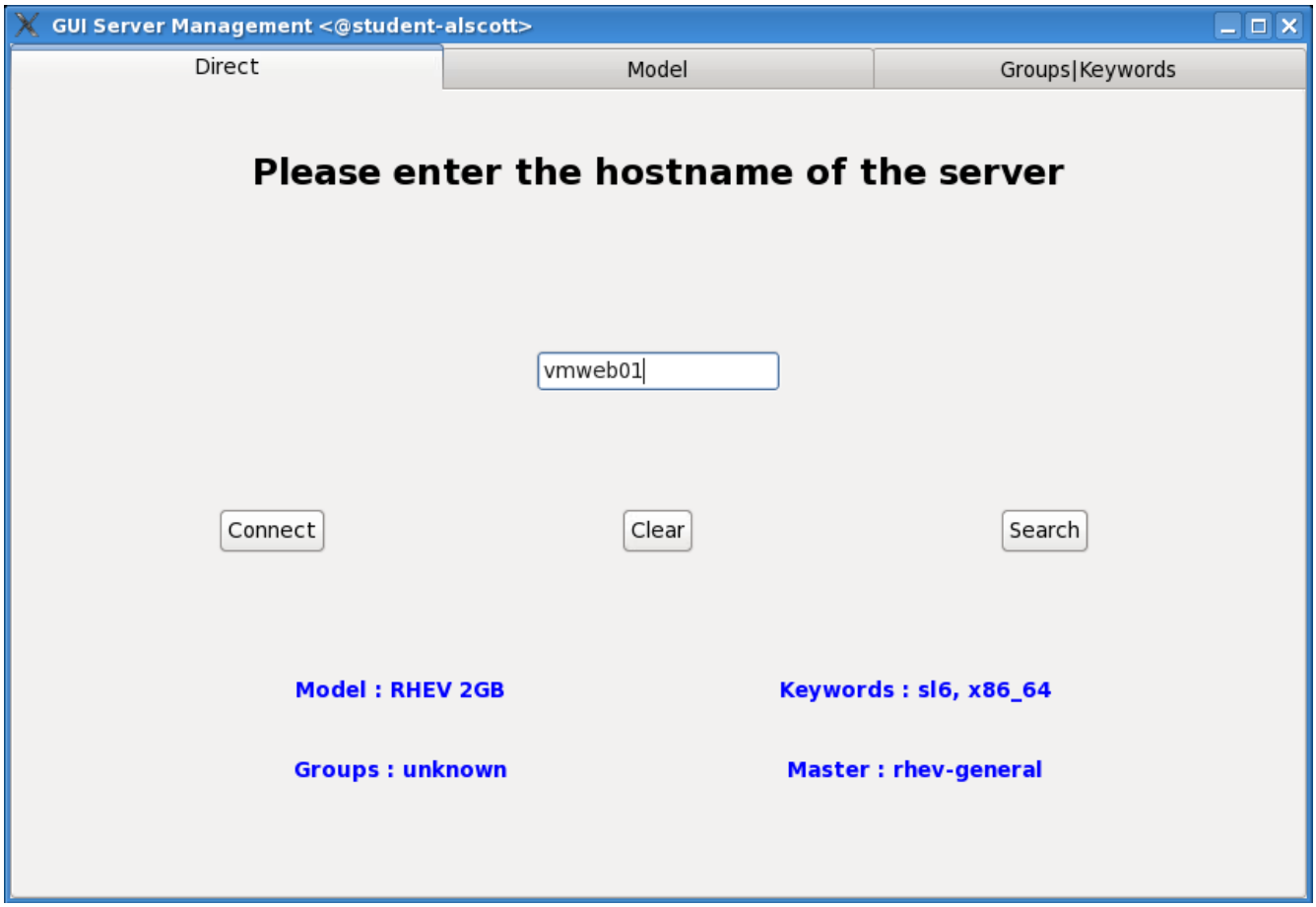

Figure 18 : Capture d'écran de l'application graphique (recherche)

Passons au deuxième onglet. Sur celui-ci l'utilisateur peut sélectionner un serveur en fonction de son modèle. Pour cela il peut se servir d'un menu déroulant permettant de choisir le modèle. A cet évènement, l'objet graphique situé sur la droite de l'écran, que l'on appelle un ''treeview'', se remplit des noms des serveurs de ce modèle, et l'utilisateur n'a plus qu'à double cliquer sur le serveur avec lequel il souhaite communiquer par console.

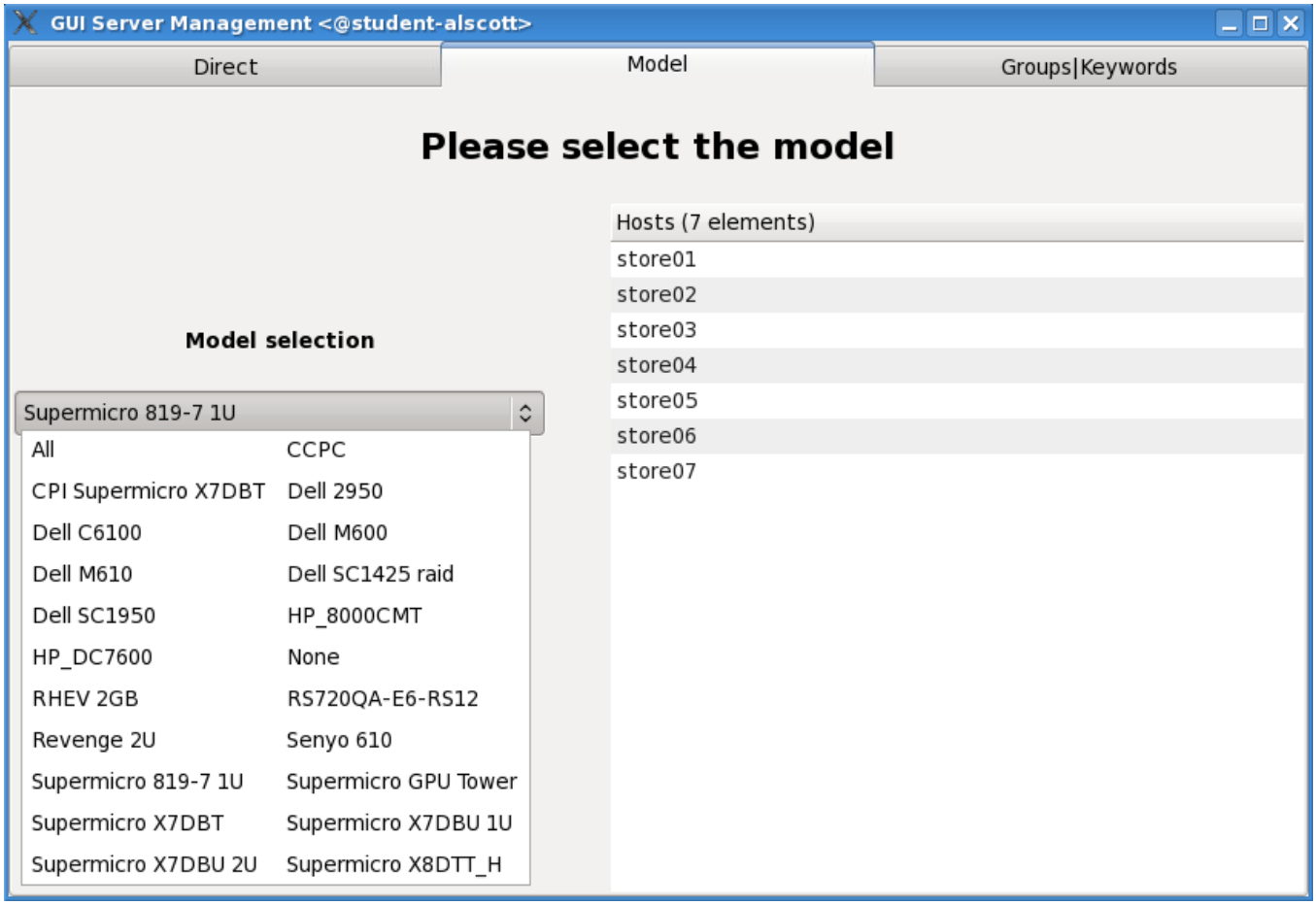

Figure 19 : Capture d'écran de l’application graphique (sélection par modèle)

Enfin le dernier onglet ressemble beaucoup à celui-ci, sauf qu’il y a deux menu déroulant. La sélection se fait donc en fonction d’un groupe et/ou un mot-clef.

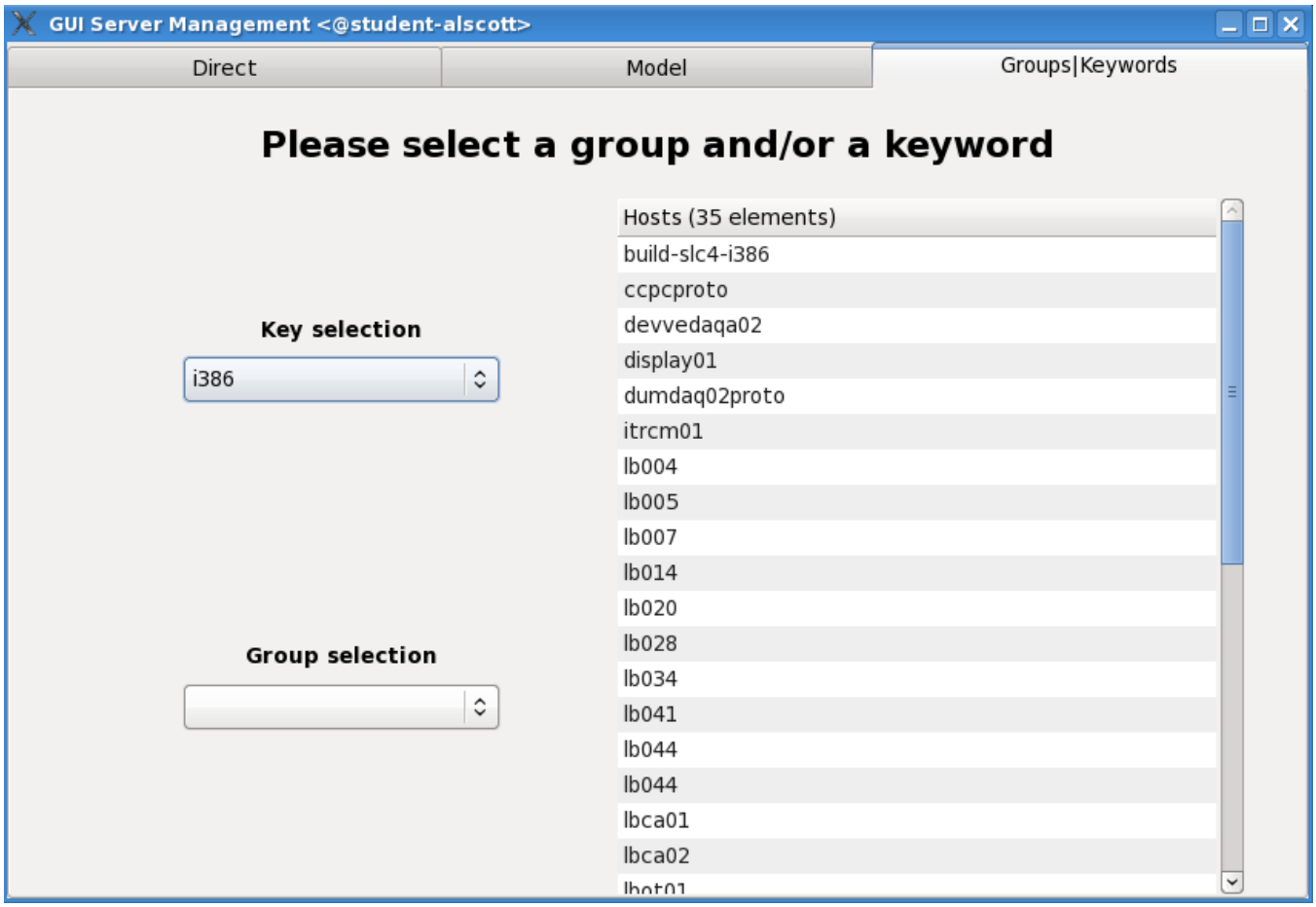

Figure 20 : Capture d'écran de l'application graphique (sélection par mot-clef)

Pour la gestion des erreurs, les exceptions remontent jusqu'à l'application graphique qui les encapsule dans une nouvelle fenêtre du type OK/Cancel.

Figure 21 : Capture d'écran de l'application graphique (exemple d'erreur)

2) Les problèmes constatés

Comme lors de tout projet, de nombreux problèmes ont été rencontrés, et certains problèmes restent insolubles.

La source principale des problèmes rencontrés est le code Java des fabricants. En effet, pour les IPMI disposant d'une interface web, les constructeurs des serveurs utilisent Java pour la redirection console. On ne peut donc pas accéder à leur code, ni même le voir. La documentation pour ce genre de développement est limitée, il n'a donc pas toujours été facile de comprendre ce qui se passait.

Le défaut le plus embêtant qui subsiste est que les serveurs Dell M600 sont injoignables. Cette erreur semble provenir du programme Java. Car même à travers l'interface web ce bug est présent. Après de nombreuses recherches, il s'avère que ce bug se déclare uniquement sur la machine virtuelle qui m'a été allouée. Et malgré des recherches dans le fichier de configuration XML de ma machine nous n'avons pu trouver la raison de ce bug. Les M600 sont progressivement retirés de la ferme de serveur pour être remplacés par des machines plus récentes (il n'y en a en fait plus qu'une dizaine qui sont répertoriés dans la base de données).

Par la suite, il s'est avéré que rarement, l'application Java ne peut pas être lancée en raison de mémoire cache corrompue, mémoire où Java stocke des informations nécessaires au bon déroulement du programme. Ce bug est reconnu par les équipes de soutiens de Java, mais pas prioritaire car il n'intervient pas dans les versions supérieures de Java et que c'est un bug assez rare.

Finalement il faut parler d'un bug dans l'API oVirt, qui est l'API permettant de récupérer les informations sur une machines virtuelles. Lorsque l'on demande des informations sur une machine en particulier, l'API nous renvoie des informations qui ne concernent pas la bonne machine. Pour contourner ce problème il faut demander la liste de toutes les machines auxquelles a accès l'utilisateur. Le problème dans ce cas, est que la constitution de cette liste prend du temps car l'API construit un objet

par machine. En moyenne cela prend 1.2 secondes à l'API pour construire un objet. Pour un utilisateur ayant accès à 25 machines virtuelles cela prendrait donc 30 secondes. Heureusement ce bug n'affecte pas les administrateurs qui sont ceux qui devraient utiliser mon programme.

Au final ces trois bugs ont un impact très limité puisque les M600 sont petit à petit remplacées, que les erreurs de mémoire caches sont rares, et que les utilisateurs principaux de mon programme sont des administrateurs.

3) Les améliorations futures

Le programme n'étant pas encore dans son cadre d'utilisation il se peut que certaines fonctionnalités non incluses dans la production se révèlent nécessaires. Dans ce cas il faudra que mon maitre de stage s'en occupe.

Par contre ce qui devrait s'avérer nécessaire dans le futur, c'est l'ajout de module pour de nouveaux modèles de serveur. Si le traitement est identique à au moins l'un de ceux que j'ai décrit, il suffira d'insérer le modèle du serveur dans la liste de modèle approprié (pour rappel IPMI avec interface, IPMI sans interface, et machine virtuelle). S'il faut au contraire développer une autre méthode d'accès, il faudra rédiger un nouveau module contenant ce code, et rajouter une liste avec ce modèle ainsi que l'appel à ce module dans le module modelList.

Conclusion

Le programme qui m'a été demandé est réalisé et fini. Il reste certes quelques bugs, mais qui relativement peu significatifs. Nous avons élargi la demande initiale, en ayant des commandes de maintenance classique directement intégrées dans mon programme.

J'ai énormément apprécié ce stage, d'une part car il m'a permis d'évoluer au sein du CERN, qui est une institution qui m'a toujours fasciné, et d'autre part, parce que j'ai beaucoup appris auprès du département online, et en particulier Loïc Brarda, sur les grandes infrastructures informatiques.

Les stagiaires de l'ISIMA au CERN ont pu visiter le détecteur LHCb ainsi que le centre de calcul du CERN. La première visite a été extrêmement intéressante car nous avons pu voir le détecteur lui-même, ainsi que les serveurs sur lesquelles je travaillais. Et la deuxième fut très intéressante au niveau informatique puisque nous avons pu voir le centre nerveux du CERN ainsi que le système de sauvegarde sur cassette.

Sans rentrer dans les détails, j'ai pu découvrir un outil de gestion qui s'appelle Quattor, qui permet de gérer un parc de machine avec ses qualités et ses défauts (surtout des défauts si on en croit les personnes entrant dans le bureau de Loïc…). J'ai également découvert la virtualisation de serveur, sur laquelle je me suis beaucoup renseigné car cela m'a passionné.

Enfin grâce à mon maitre de stage, j'ai également pu continuer de m'améliorer au niveau de la production du code, en particulier sur la rigueur.

J’espère avoir produit un travail qui aidera les administrateurs systèmes du LHCb online au quotidien. Les améliorations possibles de mon programme se verront surtout au fur et à mesure de son utilisation. Et je me tiens à disposition de mon maître de stage pour l’aider du mieux que je peux par mail.

Références bibliographiques

[1] : Le Cern en bref, Suisse, [Ressource électronique], Cern, date de dernière consultation 14/08/2013 disponible : http://home.web.cern.ch/fr/about

[2] : Etats membres, Suisse, [Ressource électronique], Cern, date de dernière consultation 14/08/2013 disponible : http://home.web.cern.ch/fr/about/member-states

[3] : Le plus grand accélérateur au monde, France, [Ressource électronique], lhc-france, date de dernière consultation 14/08/2013, disponible : http://www.lhc-france.fr/accelerateur/

[4] : The LHCb detector, Suisse, [Ressource électronique], lhcb-public, date de dernière dernière consultation : 14/08/2013, disponible : http://lhcb-public.web.cern.ch/lhcb-public/en/Detector/Detector-en.html

[5] : Learning to program using Python, Suisse, [Ressource électronique PDF], Cody Jackson, date de mise en ligne : 07/06/2013, disponible : https://docs.google.com/file/d/0B8IUCMSuNpl7MnpaQ3hhN2R0Z1k/edit

[6] : Remote Management with the Baseboard Management Controller

in Eighth-Generation Dell PowerEdge Servers, USA, [Ressource électronique PDF], Dell, date de mise en ligne : octobre 2004, disponible : http://www.dell.com/downloads/global/power/ps4q04-20040110-Zhuo.pdf

[7] : IPMI:Freight train to hell or Linda Wu & the night of the leeches, USA, [Ressource électronique PDF], Dan Farmer, date de mise en ligne : 12/08/2013, disponible : http://fish2.com/ipmi/itrain.pdf

En fait je me suis baser sur l'article précédent pour ce rapport, mais celui-ci a été retiré par l'auteur lorsque il a mis en ligne le nouveau. De plus cet article reprend les points intéressant du précédent.

[8] : ipmitool(1) manpage, USA, [Ressource électronique], Duncan Laurie, date de dernière mise à jour : 30/04/2013, disponible : http://ipmitool.sourceforge.net/manpage.html

[9] : Les avantages de la virtualisation, France, [Ressource électronique], Fabien Stéphan (plenium), date de dernière consultation : 15/08/2013, disponible : http://www.plenium.fr/prestation-informatique/projets/virtualisation-serveur/migration-monitoring-administration.html

[10] : What is a P2V conversion ?, USA, [Ressource électronique], virtualizationadmin, date de dernière mise à jour : 02/05/2008, disponible : http://www.virtualizationadmin.com/faq/p2v-conversion.html

[11] : Red Hat Launches Red Hat Enterprise Virtualization 3.2, USA, [Ressource électronique], redhat, date de mise en ligne : 12/05/2013, disponible : http://www.redhat.com/about/news/press-archive/2013/6/red-hat-launches-red-hat-enterprise-virtualization-3-2

[12] : Can hypervisor stand the test of real time ?, USA, [Ressource électronique], Yi Zheng (QNX Software Systems), date de mise en ligne : 29/04/2010, disponible : http://www.ecnmag.com/articles/2010/04/can-hypervisors-stand-test-real-time

[13] : Web Console, USA, [Ressource électronique], Mozilla, date de dernière mise à jour : 15/08/2013, disponible : https://developer.mozilla.org/en-US/docs/Tools/Web_Console

[14] : About Wireshark, USA, [Ressource électronique], Wireshark, date de dernière consultation : 15/08/2013, disponible : http://www.wireshark.org/about.html

[15] : Introducing JSON, USA, [Ressource électronique], json, date de dernière consultation : 15/08/2013, disponible : http://www.json.org/

[16] : Java Network Launcher Protocol(JNLP) support, USA, [Ressource électronique], oracle, date de derniere consultation : 15/08/2013, disponible : http://www.oracle.com/technetwork/java/javase/index-142562.html

[17] : Glade – A user interface designer, USA, [Ressource électronique], glade (gnome), date de dernière mise à jour : 06/03/2013, disponible : https://glade.gnome.org/