

Les clusters de calcul



Sébastien BADIA
Guillaume DELOURMEL
Luc DIDRY
Julien VAUBOURG

LP ASRALL
Année universitaire 2009-2010



Université Nancy 2
IUT Nancy-Charlemagne

Conventions typographiques

Toute portion de code importante sera éditée comme suit :

```
print "42\n";
```

Les chemins des répertoires et les exemples de code seront écrits avec la fonte « *Machine à écrire* » de L^AT_EX :

`/space/www/`

Les mots dont vous trouverez la définition dans le glossaire seront signalés ainsi :

[grille de calcul](#)

Les références aux livres présents dans la bibliographie seront signalés ainsi :

Exemple [\[8\]](#)

Dans la version électronique de ce rapport, les liens hypertexte, de même que les liens internes au document, sont actifs et sont donc cliquables. Ces liens se repèrent grâce à la couleur de leur police :

Une url : <http://www.grid5000.fr>

Un lien interne : [1.1](#)

Un lien vers la bibliographie : [\[3\]](#)

Citations

Les citations présentes à chaque paragraphe sont extraites de la page de l'encyclopédie en ligne *Wikipédia* sur la loi de Murphy¹.

Ces citations n'ont qu'un but humoristique.

¹http://fr.wikipedia.org/wiki/Loi_de_Murphy

Table des matières

1	Introduction	4
1	Calcul Haute Performance	4
1.1	Définition	4
1.2	Les enjeux du calcul haute performance	4
1.3	Historique [4]	5
2	Grid5000	5
2.1	Objectif de la création	5
2.2	Organisation	5
3	Objectifs	6
2	L'environnement de Grid5000	7
1	Les outils OAR	7
1.1	Réservations	7
1.2	Déploiements	8
2	Les réseaux rapides	8
2.1	Infiniband	9
2.2	Myrinet	9
3	Vérification automatique du matériel	11
1	Inventaire automatique	11
1.1	Objectifs	11
1.2	Choix des outils	12
1.3	Algorithme	13
1.4	Exemple d'utilisation	13
1.5	Relevé d'incohérences	14
1.6	Limites du script	16
2	Test des disques durs	16
2.1	Objectif	16
2.2	Le script	16
2.3	Exemple d'utilisation	17
2.4	Relevé d'incohérences	18
3	Problèmes rencontrés	20
4	Benchmarks	21
1	Préambule	21
2	MPI	21
2.1	Implémentations	22
3	Les solutions de tests existantes	22
3.1	OSU Micro-Benchmarks	22

3.2	NetPipe	24
3.3	NWS	24
3.4	Tests Alternatifs	24
3.5	Conclusion	25
4	Programme principal	25
4.1	Objectif	25
4.2	Les tests	25
4.3	L'algorithme	26
4.4	Saturation des commutateurs	29
4.5	Matrice d'ornement	29
4.6	Déploiements	30
4.7	Résultats	32
4.8	Problèmes rencontrés	36
5	Conclusion	39
1	Répartition du travail	39
2	Problèmes généraux rencontrés	39
3	Résultats obtenus	40
4	Bénéfices personnels	40
6	Bibliographie	42
	Appendices	43
1	Script de vérification automatique du matériel	44
2	README du script ivcmp	52
3	Script de benchmark des disques durs	54
4	README du script ivbon	57
5	Benchmark de Latence et Débit	58
6	Librairie Latence et Débit	74
7	Script de calcul du débit total en fonction du nombre de nœuds	75
8	Script gnuplot pour l'exploitation des résultats du script <i>bash</i>	76
9	Script Ruby d'export Yaml vers Html	76
10	Tutoriel : Support de Infiniband dans un environnement personnalisé	80
11	Tutoriel : Support de Myrinet dans un environnement personnalisé	85

Chapitre 1

Introduction

Nul n'est parfait... surtout pas les autres.

1 Calcul Haute Performance

1.1 Définition

Le calcul haute performance (en anglais [High Performance Computing \(HPC\)](#)) provient de l'utilisation d'importantes ressources de calcul, que ce soit par l'emploi de supercalculateurs, d'une [grille de calcul](#), de [clusters](#) ou d'ordinateurs ordinaires mis en réseau (à l'instar des [clusters](#), mais sans l'homogénéité matérielle de ceux-ci. Un bon exemple d'une telle structure est le projet *SETI@home*¹).

1.2 Les enjeux du calcul haute performance

Alors que l'on pourrait penser que le [HPC](#) est réservé à la recherche ou aux très grandes entreprises, de plus petites structures peuvent en avoir l'usage et les moyens. Ainsi une entreprise de taille moyenne de construction de camions a pu déterminer grâce au [HPC](#) qu'en supprimant les pare-boues de ses produits, cela permettait de réduire la résistance à l'air et d'économiser 400\$ par an et par camion en essence [3]. Si ce chiffre peut sembler ridicule, imaginez une entreprise possédant une flotte de 1000 véhicules et considérez l'économie réalisée. Depuis, cette entreprise utilise de façon régulière l'[HPC](#) pour améliorer l'efficacité du design de ses camions, qui devinrent encore plus économes.

Le [HPC](#) est aujourd'hui utilisé dans des domaines variés, allant des essais de combustion à la modélisation climatique, en passant par la simulation d'explosions nucléaires ou la recherche médicale [2]. Cette technologie permet de s'affranchir de limites telles que le temps, par exemple en proposant des simulations astronomiques, donnant des résultats en peu de temps, comparé aux centaines de millions d'années d'observation que ceux-ci auraient demandées [4]. Une autre limite que nous pouvons désormais dépasser grâce au [HPC](#) est celle de la taille : de l'infiniment petit de la fusion de l'atome à l'infiniment grand des galaxies...

¹<http://setiathome.berkeley.edu/>

1.3 Historique [4]

On peut faire remonter le calcul haute performance aux machines mécaniques de PASCAL au XVII^{ème} siècle (la *pascaline*) ou celle de BABBAGE au XIX^{ème} siècle. Puis sont apparus les ordinateurs actuels fondés sur l'architecture de VON NEUMANN (1945) et exploitant les possibilités des semi-conducteurs (1956).

Entre ces périodes, on a pu quantifier le coût d'une opération arithmétique typique de base. Ce coût a décri d'un facteur 10^{15} en passant de la première machine aux ordinateurs de 2006 dont la puissance s'exprimait en téraflops, soit 10^{12} opérations par seconde. Au milieu des années 1970, la société *Cray* développa une architecture permettant d'envisager un coût d'investissement d'un million de dollars pour un million d'opérations par seconde. Trente plus tard, les progrès technologiques des circuits électroniques et de la science de la simulation ont permis de multiplier le nombre d'opérations par un million pour un coût équivalent.

Le supercalculateur vainqueur du *top 500*² possède une puissance de plus de 1759 téraflops.

La puissance de calcul ne cesse de croître et certains pensent que nous pourrions atteindre l'exaflop³ d'ici 2015.

2 Grid5000

2.1 Objectif de la création

L'objectif de la création de Grid5000 est d'atteindre 5000 processeurs sur les différents sites. Cela permettra d'avoir les moyens d'effectuer des recherches dans le domaine des grilles. Il sera ainsi possible d'effectuer des tests réels plutôt que de ne présenter que des résultats de simulation.

2.2 Organisation

Le projet Grid5000 est réparti sur différents sites qui sont interconnectés avec le réseau [Réseau National de télécommunications pour la Technologie l'Enseignement et la Recherche \(RENATER\)](#) (voir figure 1.1, page 6).

Les sites sont composés de un ou plusieurs *clusters*, comme le site de Nancy avec Griffon et Grelon. Chaque site est géré par son propre administrateur, ce qui entraîne une hétérogénéité de configuration sur les différents sites. Par exemple, les *frontends* des différents sites qui ne disposent pas des mêmes librairies : ainsi sur Nancy nous ne pouvions pas faire l'export du résultat d'une commande au format *JSON* alors que l'option était proposée, ou encore le fait qu'on ne puisse pas accéder directement au site de Nancy et que l'on doive passer par un site comme Lyon qui permet une entrée à partir de l'extérieur du réseau.

²Projet de classification des 500 premiers supercalculateurs connus au monde.

<http://top500.org>

³ 10^{18} opérations par seconde

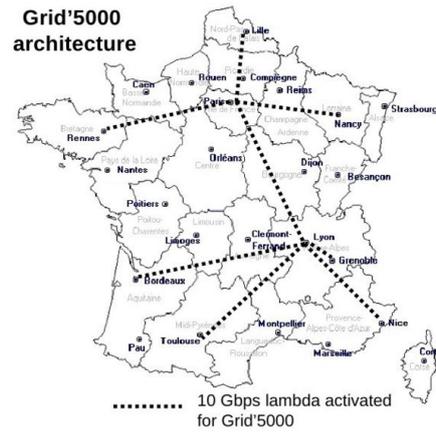


FIG. 1.1 – Répartition des sites Grid5000 sur le territoire français

3 Objectifs

L'objectif de ce projet tuteuré était de développer une suite d'outils permettant de vérifier la conformité d'un cluster acheté par rapport à l'appel d'offres auquel il fait suite et de vérifier les performances réseau et ainsi de vérifier la bonne interconnexion des nœuds.

Il s'agissait en particulier de permettre un inventaire matériel automatique des nœuds et d'effectuer un ensemble de [benchmarks](#) portant sur le stockage, le réseau en Ethernet et le réseau rapide (cartes Infiniband et Myrinet)

Chapitre 2

L'environnement de Grid5000

La caractéristique la plus constante de l'informatique est la capacité des utilisateurs à saturer tout système mis à leur disposition.

1 Les outils OAR

1.1 Réservations

OAR est une collection d'outils développée par le laboratoire IMAG¹ de l'[Institut National de Recherche en Informatique et Automatique \(INRIA\)](#) permettant de gérer un [cluster](#) de machines. Il fonctionne avec *Perl* à l'aide de *MySQL* et prend en charge le mécanisme de réservation ainsi que l'utilisation massivement simultanée des machines en [cluster](#). Grid5000 fait partie des cinq partenaires de OAR.



Lorsqu'un utilisateur souhaite accéder au projet depuis l'extérieur, il doit passer par une machine pare-feu d'un des neufs sites de France, parmi celles qui acceptent ce genre de connexion. Une fois ce pare-feu passé grâce à l'identification par clé *SSH*, c'est un frontal qui s'offre à l'utilisateur. Celui-ci est pourvu d'un espace qui lui est dédié. Cet espace se retrouvera partout où l'utilisateur ira grâce à un montage [Network File System \(NFS\)](#), dans n'importe quelle machine de n'importe quel [cluster](#), tant qu'il ne quitte pas le site. En effet, les espaces personnels ne sont pas encore partagés d'un site à l'autre, ce qui pose divers problèmes d'harmonisation et de réplication des données. Sur ce frontal, les possibilités sont très limitées et se cantonnent surtout à utiliser les outils OAR. La première chose que ces outils lui permettront de faire, c'est de réserver des nœuds. Dans le jargon des [clusters](#), cela signifie réserver un ensemble de machines physiques, pour pouvoir exécuter des actions dessus et de façon parallèle. Une réservation se fait avec son nom d'utilisateur et pour un temps donné. Il est également possible de préciser le

¹Institut d'Informatique et de Mathématiques Appliquées de Grenoble

nombre de nœuds ou de processeurs souhaités. Par le biais de filtres SQL, d'autres options sont offertes : réservation sur un [cluster](#) en particulier, sélection de nœuds possédant un matériel spécifique, etc. Si la requête trouve les machines correspondantes et que celles-ci sont libres dans l'intervalle de temps demandé, l'utilisateur est automatiquement connecté sur la première des machines de sa réservation.

Un petit tour sur les outils de monitoring des [clusters](#) (parmi eux un Ganglia, un Nagios et un diagramme de Gantt des réservations) lui indique par un rectangle de couleur le nombre de machines réservées ainsi que la durée de la réservation qui porte son nom d'utilisateur. L'environnement proposé par défaut est appelé l'environnement de développement. Celui-ci lui permettra d'exécuter des programmes qui concerneront l'ensemble des machines de sa réservation, auxquelles il peut accéder sans limite par *SSH*. Le principal intérêt de cet environnement est d'exécuter des programmes *MPI*. Nous verrons cette norme par la suite, mais retenir que c'est un moyen de faire travailler toutes les machines de la réservation en les faisant communiquer facilement.

Cet environnement de développement, différent sur chaque site et parfois même d'un [cluster](#) à l'autre, est stocké sur une partition réservée sur chacun des nœuds. L'espace de l'utilisateur est monté en *NFS* sur les nœuds réservés.

1.2 Déploiements

Le déploiement consiste à installer son propre système d'exploitation sur tous les nœuds de la réservation. Dans ce cas, la réservation doit être prise en indiquant ses intentions lors de l'appel de la commande. Des systèmes préparés sont ensuite disponibles dans la base de données des sites (qui diffère d'un site à l'autre). Il est possible d'installer un de ces systèmes, de le modifier en ajoutant ou en supprimant des programmes, des données, et de compresser le tout pour en faire un environnement personnalisé, qui pourra être déployé sur une future réservation. Lorsque l'utilisateur réserve des nœuds pour un déploiement, il est redirigé vers un environnement spécial de déploiement sur le frontal. Il pourra alors demander le déploiement d'un système sur toutes les machines, qui redémarreront sur une partition différente après avoir effectué l'installation du système. Ce déploiement est géré principalement par *kadeploy3*, un outil également développé par [l'INRIA](#).



Pour constater le redémarrage de la machine et éventuellement détecter les problèmes qui peuvent survenir à cause de l'environnement, il est possible de visualiser directement la séquence de lancement du *BIOS*, avant même que le noyau Linux ne soit démarré. *Kaconsole3* permet ceci, grâce à un contrôleur matériel et l'outil *telnet*.

2 Les réseaux rapides

Au sein du projet Grid5000, nous constatons déjà que beaucoup de choses sont hétérogènes. Mais le plus marquant est peut-être la diversité des moyens de communications au sein des [clusters](#). Selon le site ou le [cluster](#) choisi, il est en effet possible de communiquer de cinq façons différentes :

1. TCP/IP via une infrastructure Ethernet

2. TCP/IP via une infrastructure Infiniband
3. TCP/IP via une infrastructure Myrinet
4. Infiniband via une infrastructure Infiniband
5. Myrinet via une infrastructure Myrinet

2.1 Infiniband

Avec la couche standard OpenIB, Infiniband est une technologie libre et open-source supportée directement par les noyaux Linux récents. C'est un modèle de communication très performant et très économique, puisqu'il peut fonctionner sur des câbles cuivrés. Libre, il a l'avantage non négligeable d'être directement supporté par certains logiciels, directement dans les paquets de Debian.

Du fait de son implémentation libre, l'Infiniband est aujourd'hui presque incontournable dans le monde du HPC. L'implémentation dans le projet Grid5000 se fait à travers des cartes Mellanox, nommées **Host Channel Adapter (HCA)**. Celles-ci peuvent fonctionner à 10 Gb/s comme à 20 Gb/s.

Le protocole Infiniband peut être implémenté à deux niveaux différents :

- De façon native, en utilisant le protocole *VERBS*.
- Avec une surcouche logicielle comme :
 - *IPoIB* : protocole IP sur Infiniband, il permet « d'émuler » une carte Ethernet avec une carte Infiniband.
 - **SCSI RDMA Protocol (SRP)** : permet d'encapsuler des trames *SCSI* à travers l'Infiniband.
 - **Socket Direct Portocol (SDP)** : mise en place d'une couche *socket* au dessus de l'Infiniband.

Il est évident de rappeler que l'utilisation de ces surcouches au protocole Infiniband dégradent les performances. Pour des performances maximales, il faut donc utiliser de préférence la couche native *VERBS*.

Question de Requests For Comments (RFC) Le protocole Infiniband étant un standard, il est intéressant de souligner quelques RFC, dont les suivantes sont les plus intéressantes :

1. RFC4392 - IP over InfiniBand (*IPoIB*) Architecture
2. RFC4391 - *Transmission of IP over InfiniBand*
3. RFC4755 - *IP over InfiniBand : Connected Mode*

<http://www.rfc-editor.org/cgi-bin/rfcsearch.pl>

2.2 Myrinet

Ce qui n'est pas le cas de Myrinet. En effet, Myrinet est développé par la société Myricom. Dans des paquets comme ceux des implémentations *MPI*, on ne trouve donc pas le support natif de cette technologie. Pour faire du vrai Myrinet sur du réseau Myrinet, il faut donc recompiler ces paquets. Enfin, les drivers Myrinet ne peuvent s'obtenir qu'en les réclamant par courriel à la société Myricom et en justifiant d'un numéro de client. On

constate ici les problèmes liés aux technologies propriétaires qui nous ont fait perdre énormément de temps. Les infrastructures sont plus coûteuses puisque chaque nœud connecté en Myrinet doit avoir à disposition deux câbles en fibre optique, un pour les entrées et l'autre pour les sorties.

Chapitre 3

Vérification automatique du matériel

En cas de doute, videz votre chargeur.

Étant donné que la mailing list des utilisateurs de Grid5000 est souvent rédigée en anglais, nous avons jugé bon de rédiger les commentaires de nos scripts dans la langue de Shakespeare, ainsi que tous les différents messages (erreurs, usage, etc.).

Il est à noter que notre travail a évolué de façon dynamique : en effet, au fur et à mesure, M. NUSSBAUM a affiné ses demandes. Il nous était d'abord demandé de vérifier la véracité de l'OAR Database, puis de comparer les nœuds entre eux en créant des groupes partageant le même matériel. Enfin s'est greffé le [benchmark](#) des disques durs.

1 Inventaire automatique

1.1 Objectifs

Les [clusters](#) du projet Grid5000 sont achetés via des appels d'offre mais il arrive que le matériel livré ne soit pas totalement homogène : disques durs de tailles équivalentes mais de modèles différents, cartes de réseau rapide de modèles différents, etc. Le matériel peut aussi tomber en panne sans toutefois faire tomber le nœud (une barrette de ram qui cesse de fonctionner par exemple).

Comme il est difficilement envisageable pour les administrateurs de tester individuellement chaque nœud à la livraison et de façon périodique, il nous a été demandé de créer un script permettant un inventaire automatique des nœuds réservés et de les comparer entre eux pour vérifier l'homogénéité du [cluster](#) sur une liste de critères prédéfinis.

De plus, un certain nombre d'informations sur les nœuds sont entrées dans une base de données (l'*OAR Database*) lors de la mise en service du [cluster](#). L'administrateur, pour des raisons évidentes de lourdeur de la procédure, se contente de récupérer les informations d'un nœud et de les dupliquer autant de fois qu'il y a de machines dans le [cluster](#). Cela peut poser des problèmes de véracité de la base de données si le matériel n'est pas parfaitement homogène, c'est pourquoi notre script vérifie aussi la concordance à l'*OAR Database*.

1.2 Choix des outils

Langage

Nous avons choisi *Perl* comme langage de script. En effet, nous connaissions bien ce langage, le *Bash* n'était pas adapté pour les fonctionnalités que nous voulions implémenter (tables de hashage, lecture et écriture de *XML*, de *YAML*...), seul Guillaume possédait des connaissances (basiques) de *Python* et nous venions seulement de commencer les cours de *Ruby*.

Outils *Unix*

Nous avons utilisé différents outils *Unix* pour récupérer les informations que nous devons vérifier (voir tableau 3.1 page 12 pour la concordance outil <-> informations).

Si certains outils comme `uname` se sont vite imposés pour leur simplicité, le choix de `lshw` a été plus compliqué : nous avons retenu plusieurs programmes pour nous apporter les informations utiles :

- `lshw`
- `lspci`
- `ocsinventory`
- `hwinfo`

`lspci` a tout de suite été écarté en raison du peu d'informations fournies (mais il s'est révélé utile pour d'autres données).

`ocsinventory` a de même été mis hors concours mais pour sa dépendance à un serveur web et à une interface graphique pour récupérer les résultats.

`lshw` l'a emporté sur `hwinfo` grâce à sa sortie au format XML, ce qui nous a permis d'exploiter aisément ses résultats.

Outil	Information récupérée
<code>uname -nm</code>	Nom du cluster du nœud Nom (hostname) du nœud
<code>lspci</code>	Présence des cartes réseau rapide Infiniband et Myrinet Modèles des cartes réseau rapide Infiniband et Myrinet le cas échéant Interface du disque dur (sata, ide...)
<code>cat /proc/cpuinfo</code>	Nombre de cœurs des CPU
<code>lshw -xml -C disk -C memory -C network -C processor</code>	Type de cpu Quantité de mémoire vive Modèle de disque dur Taille du disque dur (ou le cumul des disques en RAID) Fréquence des CPU
<code>ifconfig</code>	Nombre de cartes ethernet activées

TAB. 3.1 – Outils *Unix* utilisés et informations récupérées

Nous avons utilisé un appel système à `ssh` pour récupérer les informations fournies par les outils *Unix*. En effet, il aurait été difficile d'utiliser le module *Perl Net* : `:SSH` : `:Perl`

car nous avons installé localement tous les modules *Perl* pour garantir la portabilité sur tous les sites et ce module ci nécessite bien trop de dépendances.

Modules *Perl*

Nous utilisons plusieurs modules *Perl*, installés dans le répertoire du script afin que celui-ci soit indépendant du site où il est utilisé.

Les modules utilisés sont :

- **YAML** : pour la récupération des informations de l'*OAR Database* en *YAML*
- **Data** : **:Dumper** : pour convertir le format *YAML* en information exploitable
- **XML** : **:Simple** : pour la récupération des informations envoyées par *lshw*
- **POSIX** " **:sys_wait_h**" : nécessaire pour la gestion des forks, il s'agit d'un module du cœur de *Perl* donc nous n'avons pas eu à l'installer

Options du script

Nous avons implémenté quelques options :

- f, --file [FILE] : spécifie le fichier contenant la liste des nœuds déployés. Par défaut, on utilisera le fichier défini par la variable d'environnement `$OAR_FILE_NODES`.
- o, --output [FILE] : permet de spécifier un fichier dans lequel écrire la sortie *YAML* (avec protection contre la réécriture)
- p, --parameter CRITERION : permet de ne comparer les nœuds que sur certains critères
- i, --install : installe automatiquement *lshw* sur tous les nœuds déployés
- r, --remove : supprime automatiquement les répertoires contenant les fichiers des informations des nœuds une fois le script terminé
- h, --help : affiche un message d'aide

1.3 Algorithme

Le fonctionnement du programme est simple :

1. Récupération de la liste des nœuds déployés par le fichier spécifié ou par la variable d'environnement `$OAR_FILE_NODES`
2. Pour chaque nœud, en forkant, nous installons *lshw* si l'option est activée et nous récupérons les sorties des outils *Unix* dans des fichiers
3. Pour chaque nœud, de façon séquentielle, nous récupérons les informations de l'*OAR Database* et des fichiers des outils *Unix* dans des tables de hashage puis nous comparons ces deux types d'informations
4. Comparaison des nœuds entre eux
5. Sortie au format *YAML* sur la sortie standard ou dans un fichier
6. Affichage des incohérences au niveau de l'*OAR Database*

1.4 Exemple d'utilisation

Nous lançons le script avec :

```
./ivcmp
```

Nous récupérons sur la sortie standard :

```

----- Notice that the printed value are from Unix tools , not from OAR database -----
1 :
  cluster : grelon
  cpuarch : x86_64
  cpucore : 2
  cpufreq : 1.6
  cputype : Intel(R) Xeon(R) CPU 5110 @ 1.60GHz
  diskmodel : ST3808110AS
  disksize : 80
  disktype : sata
  ib10g : ~
  ib10gmodel : ~
  ib20g : ~
  ib20gmodel : ~
  memnode : 2048
  myri10g : ~
  myri10gmodel : ~
  myri2g : ~
  myri2gmodel : ~
  nodes :
    grelon-102.nancy.grid5000.fr : ~
    grelon-103.nancy.grid5000.fr : ~
    grelon-104.nancy.grid5000.fr : ~
----- The Oar database cpufreq information of the node grelon-103.nancy.grid5000.fr is
not the same as we check on the node -----
      OAR database : 2          Unix tools : 1.6
----- The Oar database ethnb information of the node grelon-103.nancy.grid5000.fr is not
the same as we check on the node -----
      OAR database : 2          Unix tools : 1
----- The Oar database cpufreq information of the node grelon-104.nancy.grid5000.fr is
not the same as we check on the node -----
      OAR database : 2          Unix tools : 1.6
----- The Oar database ethnb information of the node grelon-104.nancy.grid5000.fr is not
the same as we check on the node -----
      OAR database : 2          Unix tools : 1
----- The Oar database cpufreq information of the node grelon-102.nancy.grid5000.fr is
not the same as we check on the node -----
      OAR database : 2          Unix tools : 1.6
----- The Oar database ethnb information of the node grelon-102.nancy.grid5000.fr is not
the same as we check on the node -----
      OAR database : 2          Unix tools : 1

```

Il faut noter que les messages d'incohérence relatifs à l'*OAR Database* ainsi que le message d'avertissement ne sont pas écrits dans la sortie fichier lorsqu'elle est demandée. Ces messages sont toujours dirigés vers `$STDOUT`

1.5 Relevé d'incohérences

Nous n'avons pas noté toutes les incohérences ressorties lors des tests car cela serait trop long.

Bordeaux

- bordereau-22 : ce nœud ne possédait que 3Go de mémoire vive au lieu de 4
- bordereau-4 et bordereau-7 n'ont pas le même modèle de disque dur que les autres nœuds : WDC WD800ABJS-23 au lieu de Hitachi HDS72168

Grenoble

- genepi-29 : bien que le modèle du CPU soit le même (Intel(R) Xeon(R) CPU E5420 @ 2.50GHz), `lshw` nous donne 2GHz au lieu des 2.5GHz calculés pour tous les autres nœuds.

Lille

- **cluster** Chuque : les disques durs sont de type `sata` alors que l'*OAR Database* annonce de l'`ide`. De plus il y a deux modèles différents de disques durs : WDC WD800JD-23JN et Maxtor 6Y080M0
- **cluster** Chti : deux cartes ethernet sont activées contre une annoncée
- `chti-1` : 3Go de mémoire vive contre 4Go pour le reste du **cluster**
- `chicon-14` : modèle de disque dur différent des autres nœuds : WDC WD800JD-23LS au lieu de HDS728080PLA380

Lyon

- `capricorne-55` : possède un disque de 147 Go (deux disques de 73.5Go) au lieu de 36
- `sagittaire-71, 74 et 77` : possède un disque de 147 Go (deux disques de 73.5Go) au lieu de 73

Nancy

- `griffon-91` : le modèle de carte Infiniband 10g est MT25418 alors que les autres nœuds ont une carte MT26418
- `grelon-17` : ce nœud possédait 1 Go de mémoire vive au lieu de 2
- **cluster** Grelon : les informations de l'*OAR Database* sont erronées en ce qui concerne la fréquence du CPU : au lieu des 2GHz annoncés, nous n'en relevons que 1.6. De même, le nombre de carte ethernet activées est de une carte relevée pour deux annoncées

Orsay

- **cluster** Gdx : l'information du nombre de cœurs par CPU est erronée sur l'*OAR Database* : 1 seul cœur par CPU relevé contre 2 annoncés
- `gdx-309 et gdx-310` : la fréquence du CPU annoncée sur l'*OAR Database* (2.4GHz) est erronée : 2GHz relevés
- `gdx-46` : 1Go de mémoire vive contre 2Go
- général : environ une trentaine de nœuds du **cluster** Gdx ont une fréquence de CPU différente du reste du **cluster** et il existe différents types de disques durs (environ moitié/moitié)

Rennes

- **cluster** Paramount : l'interface du disque dur relevée est de l'`ide` au lieu du `sata` annoncé. Ceci peut cependant provenir d'un problème du script
- **cluster** Paraquad : l'interface du disque dur relevée est du `sas` au lieu du `sata` annoncé.

Sophia

- **cluster** Sol : une carte ethernet est activée contre deux annoncées
- `azur-6` : nous n'obtenons pas les mêmes informations au niveau du modèle du disque dur et de son interface par rapport au reste du **cluster**

- helios-52 : nous n’obtenons pas les mêmes informations au niveau du modèle du disque dur et de son interface par rapport au reste du [cluster](#)

Toulouse

Nous n’avons relevé aucun problème à Toulouse mais ceci peut provenir du fait que nous n’avons pas pu réserver un grand nombre de nœuds pour le test (28 nœuds alors que nous avons réservé près d’une centaine de nœuds sur les autres sites)

1.6 Limites du script

Afin de pouvoir comparer certaines informations avec l’*OAR Database*, nous avons été obligé de faire des arrondis sur certains chiffres, ce qui peut entraîner des faux positifs. Par exemple, à Lyon, le [cluster](#) Sagittaire (en plus des incohérences détectées plus haut) est coupé en deux groupes : un possédant un disque dur de 73Go, l’autre un disque de 74Go. Ce genre d’erreur est facilement repérable mais pourrait poser problème dans le cadre d’un traitement automatisé des résultats.

Vous pouvez retrouver le script d’inventaire à l’annexe 1, page 44, ainsi que son README.

2 Test des disques durs

2.1 Objectif

Deux semaines avant la fin du projet, et dans la continuité de l’inventaire des nœuds, il nous a été demandé d’écrire un script pour effectuer des tests de performance sur les disques durs, car il s’agit là aussi d’une manipulation que les administrateurs ne peuvent lancer machine après machine. Cela permet ainsi de vérifier que le disque dur d’un nœud n’est pas défectueux et par exemple ne présente pas une vitesse d’écriture ou de lecture réduite par rapport aux autres nœuds.

2.2 Le script

Outils

Pour écrire ce script, nous avons utilisé le *Perl* étant donné que nous avons commencé avec ce langage pour le script de l’inventaire du matériel et que c’est celui que nous connaissions le mieux.

Pour l’accès aux nœuds réservés nous avons utilisé `ssh` qui est lancé en parallèle sur tous les nœuds. On peut ainsi effectuer les tests sur les disques durs en même temps pour que cela dure moins longtemps.

Pour effectuer les tests des disques durs, nous avons choisi d’utiliser `Bonnie++`. C’est un outil qui permet de récupérer les informations sur la vitesse d’écriture, de lecture et le nombre d’opérations effectuées par seconde sur le disque.

Pour pouvoir effectuer ces tests nous devons récupérer la taille de la mémoire ram sur chaque nœuds (nécessaire pour la taille du fichier de test). Nous utilisons pour cela l’outil `free`.

Nous avons utilisé des forks pour effectuer une parallélisation des tests, car un test seul prend déjà un temps relativement long (environ 15 minutes pour un test avec un fichier de 4G).

Algorithme

Voici le fonctionnement du programme :

- Récupération de la liste des nœuds réservés
- Pour chaque nœud en utilisant des forks :
 - On récupère la taille de la mémoire sur chaque nœud
 - On effectue le test avec la commande :

```
bonnie++ -s $ram -d /tmp -m nom_du_noeud -u root> nom_du_noeud.txt
```

- A la fin des tests nous sortons les données au format demandé.

2.3 Exemple d'utilisation

Après avoir effectué une réservation sur l'un des sites il faut lancer le script `ivbon` avec l'option `-i` pour installer `Bonnie++` lors de la première exécution. Il est ensuite possible de sélectionner le nom du fichier de sortie ainsi que son type (texte ou *YAML*).

Par exemple pour un premier lancement et pour avoir un fichier en *YAML* nommé `sortie.yml` on aura la commande suivante :

```
./ivbon -i -o sortie.yml -y
```

On récupérera alors le fichier suivant :

```
Average_speed_of_writing: 30862
Average_speed_of_reading: 66291.3
Average_number_of_operations: 205.3
grelon-33.nancy.grid5000.fr
  writing: 31076
  reading: 68539
  Number_of_operations: 207.8
grelon-34.nancy.grid5000.fr
  writing: 30361
  reading: 63508
  Number_of_operations: 208.7
grelon-35.nancy.grid5000.fr
  writing: 31149
  reading: 66827
  Number_of_operations: 199.4
```

Dans le cas où on aurait choisi une sortie texte (sans l'option `-y`) on aurait eu :

```
Average_speed_of_writing: 31579.3
Average_speed_of_reading: 66244.7
Average_number_of_operations: 197.9
      Node           | Writing   | Reading   | Number_of_operations
grelon-33.nancy.grid5000.fr | 31779    | 68636    | 204.1
grelon-34.nancy.grid5000.fr | 31197    | 63308    | 191.8
grelon-35.nancy.grid5000.fr | 31762    | 66790    | 197.7
```

2.4 Relevé d'incohérences

Bordeaux

Cluster Bordereau :

Moyenne vitesse d'écriture : 44048.9
 Moyenne vitesse de lecture : 71029.6
 Moyenne nombre d'opération : 202.7

Nœuds :	Anomalie	Valeur
bordereau-36	écriture	29208
bordereau-34	écriture	27050
bordereau-39	écriture	5982

Nœuds :	Anomalie	Valeur
bordereau-39	nb d'opérations	160.1
bordereau-23	écriture	30122
bordereau-24	écriture	28309

Lyon

Cluster Capricorne :

Moyenne vitesse d'écriture : 49777.6
 Moyenne vitesse de lecture : 55861.5
 Moyenne nombre d'opération : 450.1

Nœuds :	Anomalie	Valeur
capricorne-4	écriture	68743
capricorne-4	lecture	78586
capricorne-4	nb d'opérations	563.5
capricorne-6	écriture	34964
capricorne-6	lecture	37026
capricorne-7	écriture	38529
capricorne-7	lecture	36711
capricorne-7	nb d'opérations	640.0
capricorne-9	écriture	73872
capricorne-9	lecture	85312
capricorne-9	nb d'opérations	667.4
capricorne-15	écriture	74222
capricorne-15	lecture	85807
capricorne-15	nb d'opérations	667.7
capricorne-17	écriture	68984
capricorne-17	lecture	78815
capricorne-17	nb d'opérations	544.0
capricorne-19	écriture	35682
capricorne-19	lecture	37057

Nœuds :	Anomalie	Valeur
capricorne-21	écriture	35215
capricorne-21	lecture	37038
capricorne-23	écriture	36501
capricorne-23	lecture	36641
capricorne-24	écriture	35803
capricorne-24	lecture	37219
capricorne-26	écriture	34752
capricorne-26	lecture	36992
capricorne-28	écriture	36053
capricorne-28	lecture	37117
capricorne-35	écriture	35715
capricorne-35	lecture	36629
capricorne-35	nb d'opérations	399.6
capricorne-52	écriture	36078
capricorne-52	lecture	37177
capricorne-55	écriture	72406
capricorne-55	lecture	83734
capricorne-55	nb d'opérations	635.4

Nancy

Cluster Grelon :

Moyenne vitesse d'écriture : 36253.8
 Moyenne vitesse de lecture : 66644.7
 Moyenne nombre d'opération : 195.9

Nœuds :	Anomalie	Valeur
grelon-73	écriture	54065
grelon-74	écriture	54774
grelon-75	écriture	55113
grelon-76	écriture	53950
grelon-77	écriture	51695
grelon-78	écriture	55989
grelon-79	écriture	53547
grelon-80	écriture	56390
grelon-81	écriture	54288
grelon-82	écriture	53486

Nœuds :	Anomalie	Valeur
grelon-83	écriture	54333
grelon-84	écriture	55516
grelon-86	écriture	57512
grelon-87	écriture	55839
grelon-89	écriture	56986
grelon-91	écriture	59139
grelon-92	écriture	54436
grelon-93	écriture	53522
grelon-95	écriture	54910
grelon-96	écriture	54414

Cluster Griffon :

Moyenne vitesse d'écriture : 49786.5
 Moyenne vitesse de lecture : 77414.2
 Moyenne nombre d'opération : 162.6

Nœuds :	Anomalie	Valeur
griffon-2	écriture	34066
griffon-15	écriture	34349
griffon-23	écriture	27340
griffon-45	écriture	38831

Nœuds :	Anomalie	Valeur
griffon-85	écriture	33027
griffon-91	écriture	91811
griffon-91	lecture	106785

Rennes

Cluster Paradent :

Moyenne vitesse d'écriture : 47121.9
 Moyenne vitesse de lecture : 75675.2
 Moyenne nombre d'opération : 145.9

Nœuds :	Anomalie	Valeur
paradent-5	écriture	30237
paradent-16	écriture	32227
paradent-18	écriture	28269

Nœuds :	Anomalie	Valeur
paradent-25	écriture	33268
paradent-53	écriture	20384
paradent-56	écriture	29540

Sophia

Cluster Sol :

Moyenne vitesse d'écriture : 53964.1
 Moyenne vitesse de lecture : 64341.2
 Moyenne nombre d'opération : 222.6

Nœuds :	Anomalie	Valeur
sol-11	écriture	9068
sol-11	lecture	35121
sol-39	écriture	21749

Nœuds :	Anomalie	Valeur
sol-40	écriture	24894
sol-41	écriture	31420
sol-46	écriture	42214

Toulouse**Cluster** Pastel

Moyenne vitesse d'écriture : 41999.2

Moyenne vitesse de lecture : 58396.2

Moyenne nombre d'opération : 195.2

Nœuds :	Anomalie	Valeur
pastel-30	écriture	23820
pastel-30	lecture	39835
pastel-19	écriture	19470

Nœuds	Anomalie	Valeur
pastel-22	écriture	17584
pastel-22	lecture	39430
pastel-33	écriture	29510

3 Problèmes rencontrés

Lors de la parallélisation du script d'inventaire, nous avons voulu forker au maximum mais cela nous obligeait à partager des variables entre le processus père et ses fils. Nous avons pour cela utilisé le module IPC : `:Shareable` mais des problèmes de mémoire partagée sont apparus lors de tests à grande échelle (100 nœuds), ce qui nous a conduit à ne paralléliser que l'installation des outils *Unix* et la récupération de leurs résultats. L'extraction des données à partir des fichiers de résultats des outils *Unix* et les différentes comparaisons ne sont donc pas parallélisées.

Du 15 février au 04 mars, un bug du routeur de [RENATER](#) nous empêchait de nous connecter au site de Nancy et de récupérer notre travail par `scp`. Cela nous a posé un gros problème car nous n'avions pas eu le temps de copier le script sur les autres sites et nous avons donc commencé le rapport en attendant la résolution du bug.

Nous avons changé trois fois de format de sortie de l'outil `oarnodes` pour récupérer les informations de l'OAR Database : En effet, la sortie *XML* n'était pas véritablement homogène au niveau national : certaines informations pouvaient être en attribut sur certains sites en contenu sur d'autres. Lorsque, sur les conseils de M.NUSSBAUM, nous avons essayé de récupérer les informations en format *JSON* mais le module *Perl* correspondant n'était pas présent sur le site de Nancy. (Bug soumis aux administrateurs compétents et résolu). Nous nous sommes donc rabattu sur la sortie *YAML* pour ne pas avoir à attendre la résolution du problème.

À Grenoble, le module *Perl XML* : `:Simple` n'était pas disponible, nous avons alors dû l'installer dans le répertoire de notre script. Par précaution, nous avons fait de même pour les autres modules.

Chapitre 4

Benchmarks

S'il existe une possibilité pour qu'une expérience échoue, elle échouera.

1 Préambule

L'objectif de ces tests consiste à mesurer les débits et latences constatés entre chacun des nœuds d'un [cluster](#). Ainsi, nous mettrons en exergue l'infrastructure du réseau auquel est lié le [cluster](#) en constatant les goulots d'étranglements liés aux commutateurs, par le biais des tests de bissection. D'une façon plus générale, ils permettront de détecter les nœuds qui ne peuvent pas communiquer entre eux.

2 MPI

[Message Passing Interface \(MPI\)](#) est une norme accompagnée de bibliothèques prévue initialement pour le *C/C++* et le *Fortran* (quelques portages vers d'autres langages comme *Java*, *Python* ou *OCaml* ayant été implémentés depuis). Comme son nom l'indique, son rôle se cantonne à faire transiter des messages. Il est donc naturellement utilisé pour faire communiquer des machines entre elles au sein d'un [cluster](#), tels que ceux que propose le projet Grid5000. La spécificité de [MPI](#) par rapport aux autres bibliothèques de même rôle est d'être portable (il est prévu pour la plupart des systèmes d'exploitation), et d'être optimisé pour beaucoup de matériels. Avec ce dernier point, [MPI](#) est donc parfaitement adapté pour les réseaux rapides tels que Myrinet ou Infiniband que Grid5000 met à disposition.

Comme ses langages de prédilections le laissent supposer, la norme est ancienne (1993), et sa dernière version - [MPI 2](#) - date de 1997. C'est cette version-ci que nous utiliserons. Le langage choisi pour notre programme, lui, sera le *C* (un langage puissant et largement utilisé, pour lequel nous avons peu d'expérience et que nous avons voulu mieux maîtriser).

Historique

- **Version 1.0** : En juin 1994, le forum [MPI](#), aboutit à la définition d'un ensemble de sous-programmes concernant la bibliothèque d'échange de messages [MPI](#).

- **Version 1.1** : Juin 1995, cette version n'apporte pas de changements majeurs par rapport à la version initiale.
- **Version 1.2** : Apparue en 1997 : changements mineurs, en rapport avec la dénomination de certains programmes.
- **Version 1.3** : Septembre 2008 : clarifications sur la version 1.2 et 2.1.
- **Version 2.0** : Juillet 1997, cette version a apporté des compléments non implémentés (volontairement) dans la version 1.0 (gestion dynamique des processus, entrées/-sorties parallèles, copies de mémoire à mémoire, etc.).
- **Version 2.1** : Juin 2008 : clarifications de la version 2.0 mais aucun changements.
- **Version 2.2** : 2009 : corrections pour "coller" au standard de la 2.1.
- **Version 3.0** : 2010, cette version devrait apporter des tolérances aux fautes, des communications collectives non-bloquantes, etc.

http://meetings.mpi-forum.org/MPI_3.0_main_page.php

2.1 Implémentations

La norme [MPI](#) est parfaitement portable mais ça n'est malheureusement pas le cas de ses implémentations. En effet, même s'il est possible de compiler et d'exécuter un programme utilisant [MPI](#) sur la plupart des plate-formes, l'exécution du programme dépendra de l'implémentation utilisée.

Il existe trois principales implémentations de [MPI2](#) : OpenMPI, LAM et MPICH2. LAM est une ancienne implémentation qui n'évolue plus actuellement et qui a laissé officiellement place à OpenMPI. MPICH2, quant à lui est une autre branche indépendante. Ces deux solutions sont des logiciels libres et open-source, dont l'avantage au niveau performances semblerait revenir au second. Sur Grid5000, on trouve ces deux implémentations, ainsi parfois que LAM. Beaucoup de [clusters](#) en proposent plusieurs à la fois. Un programme *C* qui utilise [MPI](#) se compile avec le compilateur de l'implémentation choisie, et s'exécute par le biais d'une commande de gestion qui permet de sélectionner les nœuds de la réservation qui seront concernés par cette exécution.

A chaque nœud de l'exécution du programme est attribué un rang, de zéro à (*nombre de nœuds*)-1. Chaque nœud, par l'intermédiaire des commandes [MPI](#), peut consulter son numéro de rang, ainsi que le nombre de machines concernées par l'exécution du programme.

3 Les solutions de tests existantes

3.1 OSU Micro-Benchmarks

Les [benchmarks OSU Micro-Benchmarks \(OMB\)](#) sont disponibles au téléchargement sur le site de l'*Ohio State University*¹. Les tests réalisés sont très complets (latence, bande passante, etc.).

- L'exécution des tests se fait par paliers (taille du message de plus en plus grosse).
- Ces tests ne fonctionnent qu'avec deux processus, impossible donc de faire des matrices et des *bissections*.

¹<http://mvapich.cse.ohio-state.edu/benchmarks/OMB-3.1.1.tgz>

- Certains des tests sont compatibles [MPI1](#) et d'autres [MPI2](#) (MVAPICH2).

Latence

Les tests de latence utilisent spécifiquement la méthode ping-pong, et les versions bloquantes de [MPI](#) ([MPI_Send](#) et [MPI_Recv](#)). L'expéditeur envoie un message au récepteur et attend sa réponse. Le récepteur reçoit alors le message de l'expéditeur et envoie alors une réponse de même taille que le message précédemment reçu. On calcule le temps mis.

Comme toujours, plusieurs itérations du test sont effectuées dans le but de prendre la moyenne.

Lancement d'un test :

```
mpirun -np 2 --mca plm_rsh_agent oarsh -machinefile $OAR_NODEFILE ./osu_latency
```

La commande précédente nous retourne les résultats suivants :

```
# OSU \gls{MPI} Latency Test v3.1.1
# Size          Latency (us)
0                0.85
64               1.02
2048             4.08
131072           154.08
4194304          4906.56
```

Débit

Les tests de bande passante sont réalisés par l'envoi d'un message à taille fixe au récepteur. Une fois que le récepteur a reçu le message, il renvoie sa réponse sous forme d'un *Ack* (*Acknowledge*). Le processus est répété plusieurs fois, pour avoir une moyenne, et la bande passante est calculée en fonction du temps écoulé (de l'envoi du premier message, jusqu'à la fin de l'envoi du dernier message du récepteur) et le nombre d'octets envoyés.

Les fonctions [MPI](#) utilisées sont les fonctions non bloquantes d'envoi et de réception ([MPI_Isend](#) et [MPI_Irecv](#)).

L'objectif de ce test est donc de visualiser la bande passante maximale soutenue par le réseau rapide.

Lancement d'un test :

```
mpirun -np 2 --mca plm_rsh_agent oarsh -machinefile $OAR_NODEFILE ./osu_bw
```

Retour de la commande, avec la taille du mot envoyé et le débit correspondant :

```
# OSU \gls{MPI} Bandwidth Test v3.1.1
# Size          Bandwidth (MB/s)
1                1.96
[... ]
32               52.34
64              103.86
1048576          1040.28
4194304          1042.02
```

Particularités :

- Requiert deux processus maximum.
- Tests de latence et débits indépendants.

Les tests [Ohio State University \(OSU\)](#), sont très complets et pratiques, ils permettent de tester rapidement son infrastructure réseau, en utilisant [MPI](#). Cependant ils ne peuvent faire intervenir que deux machines.

3.2 NetPipe

[Network Protocol Independent Performance Evaluator \(NetPIPE\)](#) est un outil permettant une visualisation rapide des performances du réseau. Il se base sur la méthode ping-pong, avec des messages de taille croissante, entre deux processus. Les tests sont répétés plusieurs fois dans le but de constituer une moyenne.

Lancement d'un test :

```
mpirun -np 2 --mca plm_rsh_agent oarsh -machinefile $OAR_NODEFILE ./NPmi
```

Résultats :

```
0: gdx-89
1: gdx-89
Now starting the main loop
 0:      1 bytes  86569 times -->      8.31 Mbps in      0.92 usec
[... ]
102: 786435 bytes    75 times -->  6836.34 Mbps in   877.67 usec
[... ]
123: 8388611 bytes    5 times -->  6513.34 Mbps in  9825.99 usec
```

Particularités :

- Tests de débit et latence en même temps
- Incrémentation de la taille des messages par palier
- Requiert deux processus maximum
- Ne permet pas de faire de bisections ou matrices.

3.3 NWS

[Network Weather Service \(NWS\)](#) est une méthode de calcul basée sur la lecture des conditions du réseau à un instant t . Puis des méthodes mathématiques permettent de prévoir la fluctuation. C'est pour cette raison que cette méthode porte ce nom, car comme pour une prévision météorologique, on prends un cliché instantané puis on en détermine les possibles évolutions.

C'est une fraction de cette méthode que nous utiliserons pour notre programme.

3.4 Tests Alternatifs

Différents autres outils de base peuvent être utilisés, dans le but de tester les performances.

Cependant, leur utilisation est souvent plus complexe.

- Les commandes de la carte (`mx_info`, `ibv_devinfo`, `ib_read_lat`, etc.)
- La suite Kanif (`Kash`, `Kaput`, etc.)
- `Perfquery`
- `Perftests`

3.5 Conclusion

Malgré les nombreuses solutions testées, nous n'avons pas trouvé de solution existante convaincante, permettant de mettre en avant la faiblesse d'un nœud spécifique dans un `cluster`. De plus la quasi-majorité des solutions ne mettait en œuvre que deux nœuds, impossible donc de faire un test de débit/latence sur tout un cluster, ou même de faire une bisection ainsi qu'une matrice.

C'est pour cela que nous avons été amené à développer notre propre programme de test.

4 Programme principal

4.1 Objectif

L'objectif du programme est de mesurer la latence et le débit entre chaque paire de nœuds possible. Si une réservation concerne dix nœuds, il y a aura donc dix fois dix tests de débit/latence à faire. Le programme doit présenter les résultats sous forme d'une matrice, et avoir la possibilité d'exporter ces résultats sous forme de fichier *YAML*.

Un mode bisection est également à prévoir : il s'agit de couper en deux la liste de tous les nœuds pour créer des paires. Une fois ces paires formées, tous les tests doivent s'exécuter en même temps afin d'éventuellement saturer le commutateur réseau. C'est ensuite le débit total de tous ces échanges qui sera utilisé pour faire l'interprétation.

4.2 Les tests

Pour la latence, nous considérerons qu'il s'agit du temps mis par un message vide pour transiter d'un nœud à l'autre. Elle sera donc obtenue en calculant le temps mis depuis l'envoi du message au nœud distant jusqu'à la réception de son acquittement. Puisqu'il s'agit de ne calculer que le temps de transition du message, ce temps doit être divisé par deux. La latence se mesure en microsecondes.

Pour ce qui est du débit, nous considérerons qu'il s'agit du temps mis par un octet pour transiter d'un nœud à l'autre. Pour le calculer plus précisément et éliminer les imperfections du réseau, il est nécessaire d'envoyer des messages de plusieurs octets (nous considérons qu'il est nécessaire d'avoir plus de soixante-quatre kilo-octets pour avoir des résultats pertinents) pour ensuite diviser le temps constaté. Elle sera donc obtenue en calculant le temps mis depuis l'envoi du message jusqu'à la réponse - vide - du nœud distant. La latence calculée auparavant sera retirée deux fois (aller et retour) afin de n'obtenir que le temps consacré pour faire transiter le message, sans tenir compte de l'impureté du réseau. Le débit se mesure en méga-octets par seconde.

Cette façon de calculer ces deux valeurs relève de l'approche *NWS*².

²<http://nws.cs.ucsb.edu>

4.3 L'algorithme

Principe

Notre programme exploite au mieux les possibilités offertes par la norme [MPI2](#). Ainsi, nous avons créé de nouvelles structures [MPI](#) à partir de structures C que nous utilisons dans notre programme. En utilisant les types créés, nous pouvons passer ces structures d'un nœud à l'autre. Trois structures sont au rendez-vous : `Bench`, `YourTest` et `MyResult`. `Bench` est une structure qui contient un numéro d'envoyeur (correspondant à un rang), un numéro de receveur (un autre rang) ainsi que deux valeurs correspondant au débit et à la latence constatés lors du test consistant à aller de l'envoyeur vers le receveur. `YourTest`, elle, contient un attribut `role` et un numéro de rang. Le rôle peut correspondre soit à *envoyeur*, *receveur* ou *désactivé*. Nous verrons ce dernier dans la suite de l'explication du programme, et les deux autres sont maintenant explicites. Enfin, `MyResult` contient un attribut de type `Bench` et une chaîne de caractères correspondant à un nom de domaine.



Le programme se divise en deux contextes principaux : le nœud de rang zéro, et les autres. En effet, le nœud qui se verra attribuer ce rang spécial aura pour charge de centraliser les résultats, mais aussi de créer les différents tests, les attribuer aux nœuds, et donner le départ des tests (de façon globale si c'est une bisection, pair à pair sinon). Ce nœud ne participera donc pas aux tests, et ne se retrouvera pas dans les résultats. Si on veut que la machine de ce rang-ci participe, il faut la répéter deux fois en début du fichiers des nœuds participants. [MPI](#) est multi-processeurs, il prendra donc le premier processeur du nœud de rang zéro (le premier dans la liste) pour gérer la batterie de tests, et le second pour représenter la machine au sein des résultats.

Le contexte du nœud zéro, que nous appellerons dorénavant maître, commence son exécution en différenciant une exécution pour un mode bisection (option `-b` du programme) ou non. Les autres nœuds, eux, se mettent à l'écoute du maître. Selon que le mode bisection est actif ou non, ils écouteront de façon collective (point qui sera abordé par la suite) ou individuellement. Écouter signifie se mettre en attente d'un envoi du maître, et bloquer le programme jusqu'à ce que celui-ci ait daigné leur envoyer un message.

Les sources du programmes sont disponibles en annexe [5](#) et [6](#) pages [58](#) et [74](#).

Le mode matrice

Pour ce mode, chacun des nœuds qui n'est pas maître commence par se mettre en écoute, individuellement, sur celui-ci. Ceci signifie que leur exécution ne continuera que lorsque celui-ci leur aura explicitement envoyé un message. Ce qu'ils attendent tous, c'est un test de type `YourTest`. Une fois qu'ils auront ce test, ils pourront consulter leur rôle et savoir quelle attitude adopter dans la suite de leur exécution.

Le maître, lui, parcourt les rangs des nœuds un par un. Pour chacun de ces nœuds il se livrera au même rituel : il commence par invoquer la préparation des tests, en fonction du rang sur lequel il s'est arrêté. La préparation consiste à parcourir tous les rangs qui ne sont pas le rang donné. Pour chacun, un test est créé : ils seront tous receveurs, et leur compagnon de jeu sera le rang du nœud donné. Tous ces tests leur sont ensuite envoyés individuellement. Chacun était à l'écoute de son test, et se débloque donc à cette arrivée. Comme ils sont tous de même rôle, ils choisissent tous la même action : se mettre à écouter le rang donné. Ce dernier est le seul à ne pas avoir de rôle dans cette partie, nous lui donnons donc naturellement le rôle d'envoyeur. Une fois qu'il a réceptionné son rôle, il sait exactement ce qu'il doit faire en fonction. Il commencera par déclencher un chronomètre, puis enverra un message vide au premier receveur, et se mettra en écoute de celui-ci. Le receveur s'empressera de répondre et l'envoyeur pourra arrêter son chronomètre. Un calcul de latence est alors déduit du temps mis entre les deux mesures de temps. Dans la foulée le second test se déroule de la même façon, mais avec un message bien plus gros (selon la valeur de l'option `-s` passée au programme, et par défaut un mégaoctet). Le receveur renverra sa réponse - vide - et l'envoyeur pourra valider ce nouveau chronomètre. En fonction du temps mis par ce second test et de la latence calculée précédemment, on déduit le débit. Ces deux tests seront répétés un certain nombre de fois (selon l'option `-p`, et par défaut dix fois). Un débit ainsi qu'une latence seront déduits de la moyenne de ceux-ci. En fonctionnant ainsi, nous éliminons en partie les défauts du réseau qui peuvent intervenir, et qui ne pourrait pas rendre un seul test représentatif.

Ces résultats sont naturellement stockés dans une structure de type `MyResult`. En plus de contenir les informations concernant les parties des tests ainsi que les résultats, un nom de domaine est fourni. Il s'agit du nom de domaine de l'envoyeur, qui communiquera cette structure au maître (qui s'est mis en écoute d'une donnée de ce type de la part de l'envoyeur actuel, juste après avoir donné le rôle). Le maître reçoit ce résultat, le stocke, et reformule un test à l'envoyeur pour produire des résultats avec le second receveur. Le premier receveur, lui, s'est remis en écoute du maître. Dans ce mode, tous les nœuds se remettent en écoute du maître après avoir fini jusqu'à ne plus avoir de rôle à jouer (si `nb` représente le nombre de nœuds concernés par l'exécution du programme, chaque nœud sera $nb-2$ fois envoyeur et autant de fois receveur ; les deux nœuds retirés chaque fois étant le nœud de rang zéro qui fait office de maître ainsi que lui-même).

Une fois que chaque nœud a réalisé un test avec chacun des autres nœuds et qu'il a remonté ses résultats au maître, le programme se fini pour eux, en fermant la connexion `MPI` et en quittant le programme. Le maître se retrouve seul pour rendre compte des résultats. Il les organise et les affiche sous forme de matrice sur la sortie standard. Il fournit en plus des statistiques telles que la paire de nœud qui a enregistré le plus bas et le plus haut débit, et le même type de résultats pour la latence. Si l'utilisateur a demandé une sortie `YAML`, celle est effectuée en plus, dans le fichier donné en paramètre (option `-o`).

Le mode bisection

Si l'utilisateur a demandé le mode bisection (option `-b`), la philosophie des tests diffère. Il s'agit en effet de ne plus distribuer les rôles, de récupérer les résultats et surtout de faire les tests consécutivement, mais d'exécuter toutes ces tâches parallèlement, en s'assurant que tous démarrent en même temps. L'objectif est en effet de saturer le com-

mutateur réseau et de constater quel est le débit total lorsque toute une moitié de `cluster` échange une information avec l'autre moitié. Avec ce type de test, nous devrions mettre en évidence l'emplacement des commutateurs, puisqu'ils devraient agir en goulot d'étranglement, ce qui se reflétera sur les débits constatés entre certains nœuds.

A l'exécution du programme, tous les nœuds qui ne sont pas le maître se mettent à l'écoute d'un test de celui-ci. Mais d'une façon différente du mode matrice, puisque nous exploitons ici l'autre force `MPI` : les fonctions collectives. Ainsi, chaque nœud se met à appeler précisément la même fonction qui stipule qu'ils recevront un message de type `YourTest` du maître. Chacun de ces nœuds ne sera débloquenté que lorsque le maître lui-même aura invoqué cette même fonction, pour envoyer ses tests.

Le maître, lui, commence comme pour le mode matrice : il s'attelle à préparer les tests. Cette préparation consiste avant tout à créer des couples. Pour faire une bonne bisection, il faut d'abord couper la liste des nœuds en deux parties égales (en prenant soin d'écartier le nœud maître ainsi que le petit dernier si jamais le nombre était impair). Pour prendre des nœuds deux à deux et créer des couples d'envoyeur/receveur, il existe deux écoles : l'école de l'aléatoire et l'école de *la moitié+1*. L'une ou l'autre de ces écoles est utilisée selon que l'option `-rb` (pour *Random Bisection*) ait été passée. Ainsi, dans le mode aléatoire, tous les nœuds n'ayant pas de rôle seront rassemblés dans un grand panier. Puis, tour à tour, deux seront tirés au sort et se verront respectivement s'attribuer le rôle d'envoyeur ou de receveur de l'autre, jusqu'à ce que tous les nœuds aient été intégrés dans un test. Dans le mode *moitié+1*, le premier nœud de la liste sera envoyeur, avec le premier nœud de la seconde moitié de la liste, plus un, qui sera receveur. Ainsi de suite, jusqu'à ce que le dernier nœud de la première moitié ait été intégré à un test.

Vient alors le temps de distribuer ces rôles : le maître se décide enfin à appeler la fonction collective sur laquelle tous les autres nœuds sont bloqués, impatients de recevoir leurs tests, comme un enfant sous un sapin de Noël qui regarderait inlassablement le conduit de la cheminée. Une fois que chacun d'eux a reçu son rôle, chacun se met en position de combat : la même que si ils étaient en mode matrice, sauf qu'il se mette d'abord à réécouter le maître. D'une façon différente cette fois-ci : à l'aide d'une seconde fonction collective correspondante à un broadcast. Tout le monde recevra donc en même temps le même message, et ce sera un message vide.

L'intérêt de ce message qui bloque le début des tests est de s'assurer que chaque test commencera exactement au même instant, et qu'il n'y aura pas un décalage dû au temps pour chacun de recevoir son rôle (puisque tous les nœuds ne sont pas accessibles à la même vitesse, ce qui est justement ce que nous cherchons à prouver). Le message vide est envoyé, les envoyeurs envoient et les receveurs reçoivent. Les tests se déroulent de la même façon que dans le mode matrice à une exception près : chaque test devant être refait un certain nombre de fois pour prendre la moyenne de tous, il ne s'agit pas que les paires de nœuds les plus rapides terminent tous leurs tests avant les autres. Si une telle chose se produisait, les goulots d'étranglement n'étrangleraient plus, et les derniers nœuds communiqueraient plus rapidement sur leurs derniers tests, faussant ainsi leurs propres moyennes de résultats. Pour pallier ce problème, chacun des nœuds se remet en écoute de la fonction de broadcast du maître avant de relancer un test. Ainsi, les plus rapides attendent toujours les plus lents avant de recommencer leurs tests.

Une fois que chacun a fini sa série de tests avec son partenaire, chacun appelle la dernière fonction collective, inverse exact de la première utilisée. C'est à dire que ça n'est plus le maître qui envoie simultanément tous les tests personnalisés aux nœuds, mais les nœuds qui envoient chacun leur résultat au maître. Celui-ci les reçoit tous, les stocke, et les gère de la même façon pour le mode matrice.

4.4 Saturation des commutateurs

Une utilisation pratique du mode bisection consiste à réaliser une série de tests consécutifs, en faisant varier le nombre de nœuds dans la partie. Ainsi, le programme sera d'abord exécuté avec trois nœuds (option `-np` de `mpirun`). De cette bisection, le débit total sera récupéré (addition des débits constatés entre chaque paire de nœuds). Le programme sera réexécuté en ajoutant quelques nœuds et ainsi de suite jusqu'à tous les utiliser. De ces expériences ressortent des relations entre le nombre de nœuds impliqués dans une bisection et le débit total constaté.

Nous pouvons ainsi constater le seuil de saturation du réseau lorsque les commutateurs ne tiennent plus suffisamment la charge pour accueillir encore plus de débit et agissent donc en goulot d'étranglement. Grâce à l'option `-g` du programme, seul le nombre de nœuds en fonction du débit total de la bisection sont affichés. Ainsi, il est possible de compléter au fur et à mesure un fichier qui bénéficiera d'une syntaxe compatible avec le programme `gnuplot` qui permettra de tracer des graphiques de débit. Un script `bash` a été écrit pour automatiser cette série d'expériences et créer automatiquement le fichier `gnuplot`. Sa seule option est un pas correspondant au nombre de machines à ajouter à chaque nouvelle bisection.

La source du script est disponible en annexe 7 page 75.

4.5 Matrice d'ornement

Enfin, un script *Ruby* (langage moderne et très bien interfacé avec le *YAML*) a été écrit pour exporter la matrice des résultats sur un site internet. Ainsi, à partir d'un fichier *YAML* généré par le programme, le script crée un fichier *HTML* de la matrice, plus esthétique et plus facilement consultable. De plus, les débits et latences constatés sont colorés en fonction de leur position dans le classement des performances : le débit et la latence les plus bas sont marqués sur fond rouge, et les plus hauts sur fond vert.

De façon intermédiaire, les débits et latences faisant partie du premier quartile des résultats sont marqués en vert, et ceux faisant partie du troisième quartile sont en rouge. Un quartile est une valeur statistique mathématique qui se calcule en classant tous les résultats, pour en calculant pour chaque « l'effectif cumulé croissant » (*ecc*) qui leur est associé (addition de tous les résultats qui les précède). Lorsque l'*ecc* d'une valeur est inférieure à un quatrième du total des résultats, alors celle-ci fait partie du premier quartile. Lorsqu'elle fait partie du quatrième quart, il s'agit d'une valeur du dernier quartile. Ce code couleur permet de repérer instantanément les nœuds qui font défaut et ceux qui sont performants. Ces derniers étant souvent proches les uns des autres ou font partie d'un commutateur peu chargé.

-i, -yaml	Fichier <i>YAML</i> d'entrée.
-o, -html	Fichier HTML de sortie.
-h	Une aide de ce type.

La source du script est disponible en annexe 9 page 76.

4.6 Déploiements

L'intérêt

Lors de l'utilisation d'un [cluster](#), on peut être amené à utiliser des logiciels qui ne sont pas installés sur les environnements de production. L'intérêt des déploiements est alors multiple.

Dans un premier temps, il permet de configurer chaque nœud à sa guise, installer les logiciels que l'on veut, et ainsi ne pas « polluer » l'environnement de production avec des logiciels dont nous seuls avons l'utilité. Il permet également de changer la configuration même du système (mise à jour du noyau, déploiement avec des distributions exotiques, etc.).

Dans un second temps le déploiement permet de sauvegarder et d'exporter l'environnement ainsi modifié. Puisque le déploiement se fait à partir d'une archive, l'archive peut être portée sur différents sites, et ainsi être réutilisée ultérieurement.

Dans un dernier temps, le déploiement permet de lisser l'exécution de nos programmes, principalement d'un site à l'autre puisque l'hétérogénéité des environnements de production empêche le fonctionnement normal. L'utilisation d'un même environnement permet donc de lisser ces problèmes.

L'utilisation de [MPI](#) dans les environnements de développement est assujettie aux problèmes. En effet, selon les [clusters](#), on rencontre différentes implémentations de [MPI](#), qui ne supporteront pas les mêmes options. De plus, pour les [clusters](#) concernés, certains ne supportent pas par défaut leurs propres réseaux rapides. Enfin, beaucoup d'environnements soulèvent des problèmes de connexions entre les nœuds. Sur certains [clusters](#), [MPI](#) devra contacter les autres nœuds via `ssh`, d'autres via `oarsh`, ou encore via `rsh`. Certains nécessitent l'utilisation d'une clé *SSH* sans *passphrase* ou un `user-agent` pour se connecter sans demander de mot de passe (qui serait impossible à renseigner).

L'intérêt de déployer son propre environnement est de faire face à tous ces problèmes en recréant un système qui supportera parfaitement les deux types de réseaux rapides et qui proposera une seule implémentation de notre choix de [MPI](#) (qui fonctionnera avec les réseaux rapides). Pour les interconnexions, une liaison *SSH* par clé *SSH* sans *passphrase* sera largement suffisante.

Support de Infiniband

Le support d'Infiniband sur un environnement fraîchement installé est relativement facile à mettre en place. Son aspect libre y étant pour beaucoup !



En effet les cartes Infiniband Mellanox installées sur les différents nœuds de Grid5000 sont déjà reconnues par le noyau Linux. L'étape la plus compliquée consiste alors à lister les modules utiles, pour ce que l'on veut faire (support de IP sur Infiniband) et de les activer (`modprobe <module>`), ou de les inscrire directement dans le fichier `/etc/modules`. Ainsi, ceux-ci seront directement chargés au prochain déploiement.

Nous sommes donc partis d'un système *lenny-x64-base*, que nous avons mis à jour afin de bénéficier d'un noyau plus récent que le 2.6.26. Avec la mise à jour, nous avons installés les drivers propriétaires (*sic*), et OpenMPI, une implémentation libre de [MPI](#).

Certaines modifications ont été apportées, comme l'ajout d'utilisateurs, la mise en place de clés et la cohabitation entre Infiniband et Myrinet.

Un tutoriel complet créé par nos soins est disponible en annexe 10 (page 80).

Support de Myrinet

Le support de Myrinet sur l'environnement personnalisé n'aura pas été tout à fait aisé.



Un tutoriel complet créé par nos soins est disponible en annexe 11 (page 85).

Système de départ Pour partir d'un bon pied, nous avons utilisé un des environnements proposés dans la base de données des environnements de OAR. Comme le seul vraiment supporté est la version lenny de Debian, c'est celle-ci que nous mettons en œuvre. Une fois déployée, la première étape consiste à la mettre à jour : nous passons le système en testing. Un problème au niveau de `udev`, récurrent à ce moment là, intervient. Il faut redémarrer le système en plein milieu de l'installation pour la poursuivre.

Drivers Myrinet Pour que les drivers Myrinet fonctionnent, il faut qu'ils soient compilés avec précisément le même compilateur que le noyau du système. La seule façon de s'en assurer est donc de recompiler le noyau, en en profitant pour installer une version récente (2.6.32). Pour bien faire les choses, nous créons un paquet Debian de ce nouveau noyau.

Vient la récupération des drivers Myrinet : lorsque nous tentons de les récupérer, nous découvrons que la société Myricom impose l'envoi d'un email pour obtenir le mot de passe permettant le téléchargement. Une fois cette contrainte exécutée, nous recevons

une réponse nous demandant de fournir un numéro de client³. Ce genre de chose est laborieuse à obtenir, notre tuteur réussi finalement à retrouver un email contenant le mot de passe, nous permettant toujours de télécharger les sources des drivers. Par la suite, nous tenterons aussi de récupérer une version de MPICH2 (une implémentation de MPI) qui supporte le Myrinet, qui se trouve sur le même site Internet. Là encore, nous serons confronté au même problème, sans solution cette fois-ci (sauf celle d'utiliser une autre implémentation de MPI à laquelle nous intégrerons ce support). J'en profite donc pour mettre l'accent sur les désagréments des logiciels propriétaires, soumis à des contraintes stupides, et surtout consommatrices de temps. Le support d'Infiniband, l'autre réseau rapide, est libre, et donc supporté par défaut dans les noyaux Linux récents.

Une fois les drivers finalement récupérés et compilés, un second paquet Debian a été créé. Pour lancer la création de l'interface de Myrinet au démarrage de la machine (et donc à la fin du déploiement), nous avons écrit un service qui lance et arrête l'ensemble des commandes qui activent le support de Myrinet.

Enfin, nous avons fait le choix de OpenMPI pour les support de MPI sur nos environnements. Ce choix par rapport à MPICH2 s'est fait parce que celui-ci ne supporte pas Infiniband par défaut et parce que la version Myrinet ne nous est pas accessible. Puisque Myrinet est propriétaire, le paquet Debian ne le supporte pas. Nous l'avons donc recompilé et créé par la même occasion un paquet Debian fonctionnel.

Une fois un utilisateur créé, des clés *SSH* bien placées et avoir vérifié que tout fonctionne bien, nous avons enregistré ce nouvel environnement, prêt à faire communiquer notre programme MPI sans problème et en utilisant éventuellement les réseaux rapides.

Problèmes Malheureusement, tout n'est pas si parfait. En effet, comme nous avons déjà eu l'occasion de nous en rendre compte, l'universalité des clusters est une illusion. Ainsi l'image que nous avons créé à Orsay ne fonctionne pas dans certaines villes (Rennes, par exemple). Certains problèmes ont été résolus par l'installation de *firmwares* propriétaires (*sic*). D'autres restent toujours obscurs. Nous avons en effet dialogué avec les utilisateurs de la liste email à laquelle nous nous sommes tous abonnés, et à ce jour nous ne savons toujours pas pourquoi l'image refuse de se déployer sur ces villes.

Dans l'avenir, nous espérons que tous les environnements de développement proposent des implémentations de MPI fonctionnelles, avec des interconnexions faciles, et un support natif de leurs propres réseaux rapides. Durant la période de notre projet tuteuré, des avancements ont déjà été effectués dans ce sens sur plusieurs sites.

4.7 Résultats

Utilisation du programme

Le programme s'utilise à travers la commande `mpirun`, en indiquant un fichier de nœuds dédoublonné. Éventuellement répéter le premier nœud (qui sert de maître) pour

³« Please send a serial number from one component - the 6-digit number prefixed by "SN" on the white label - so that we can verify your license »

le faire apparaître dans les résultats.

- s <n>K** Taille des messages utilisés pour les tests de débit en octets (avec les suffixes K, M ou G).
Taille minimum de 64K, par défaut 1M.
- p <n>** Précision des tests (nombre de fois que chaque test sera répété pour prendre la moyenne de tous).
Par défaut 10.
- b** Mode bisection (premier nœud de la première moitié avec le premier de la seconde, et ainsi de suite).
- rb, -r** Mode bisection, avec des formations de paires de nœuds aléatoires.
- bg, -g** Sortie pour un fichier de données gnuplot.
- o <file>** Sortie *YAML*.
- h** Une aide de ce type.

Exemples de sortie

Matrice de débits/latences avec cinq nœuds concernés :

	To gdx-212	To gdx-213	To gdx-214	To gdx-215
From gdx-212		14,400 us 635,740 Mo/s	15,128 us 644,675 Mo/s	14,949 us 640,825 Mo/s
From gdx-213	7,844 us 671,236 Mo/s		12,708 us 669,531 Mo/s	12,767 us 665,154 Mo/s
From gdx-214	7,665 us 668,970 Mo/s	7,403 us 669,591 Mo/s		13,196 us 662,985 Mo/s
From gdx-215	7,200 us 672,025 Mo/s	7,010 us 673,011 Mo/s	7,057 us 670,242 Mo/s	

Latency :

Max : 15,128 us From gdx-212 to gdx-214
 Min : 7,010 us From gdx-215 to gdx-213
 Sum : 127,327 us
 Avg : 10,611 us

Flow :

Max : 673,011 Mo/s From gdx-215 to gdx-213
 Min : 635,740 Mo/s From gdx-212 to gdx-213
 Sum : 7943,985 Mo/s
 Avg : 661,999 Mo/s

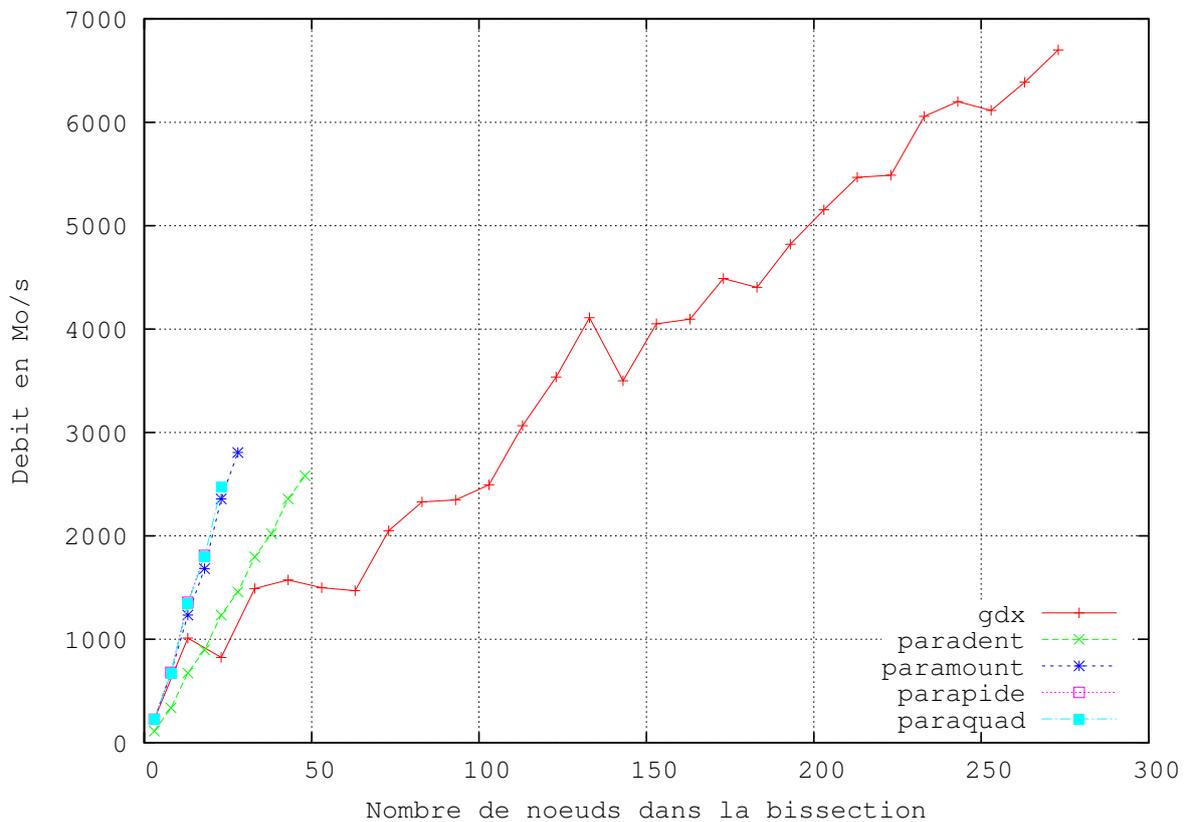
Bisection, avec six nœuds (option *-b*) :

Sortie *YAML* (option *-o*) :

```

gdx-212 :
gdx-213 :
  latency : 14,067
  flow : 633,161
gdx-214 :
  latency : 14,400
  flow : 646,889
gdx-215 :
  latency : 14,949
  flow : 633,000
gdx-216 :
  latency : 15,390
  flow : 641,042
gdx-213 :
gdx-212 :
  latency : 7,343
  flow : 668,495
gdx-214 :
  latency : 11,909
  flow : 677,445
gdx-215 :
  latency : 11,206
  flow : 663,127
gdx-216 :
  latency : 12,112
  flow : 671,858
gdx-214 :
gdx-212 :
  latency : 7,880
  flow : 670,927
gdx-213 :
  
```

Tests consécutifs de bisections avec l'option *-g* et le script *bash*, pour des débits sur Ethernet (graphique avec le script *gnuplot* disponible en annexe 8) :



Sortie HTML de la matrice avec le script *ruby* :

Latency flow tests
(with nice colors)

	netgdx-8	netgdx-9	netgdx-30	netgdx-4	netgdx-5	netgdx-7	
netgdx-8		159.396	156.104	156.758	157.174	156.814	MB/s
		83.458	64.325	60.236	58.532	60.856	µs
netgdx-9	154.599		155.611	157.134	153.533	155.683	MB/s
	60.415		64.945	62.573	58.591	59.39	µs
netgdx-30	173.417	174.385		151.102	172.868	170.378	MB/s
	79.823	80.562		74.387	74.649	76.795	µs
netgdx-4	179.281	178.96	157.702		175.065	173.463	MB/s
	76.592	77.105	63.217		75.09	77.2	µs
netgdx-5	180.827	158.725	155.13	156.474		158.213	MB/s
	77.176	78.905	64.719	61.488		78.297	µs
netgdx-7	179.638	159.51	155.246	157.78	156.128		MB/s
	80.955	82.481	62.609	61.405	58.663		µs

4.8 Problèmes rencontrés

MPI

La majorité des problèmes rencontrés avec [MPI](#), relève de l'hétérogénéité des différents sites, ainsi un programme ne se lancera pas de la même manière sur les environnements de production, selon que l'on soit à Rennes, à Nancy, ou autre part.

Erreur concernant un réseau rapide Myrinet alors que le [cluster](#) (netgdx sur Orsay, ainsi que certains [clusters](#) de Rennes) n'en propose pas : *warning :regcache incompatible with malloc.*

Il semblerait que le problème vienne de la compilation de OpenMPI (compilation pour utiliser Myrinet, alors que c'est souvent Infiniband qui est supporté).

Impossible de faire communiquer les programmes entre eux à Lyon, une invitation de mot de passe impossible à saisir bloque les accès.

```
$ mpirun -np 5 --mca plm_rsh_agent oarsh -machinefile $OAR_NODEFILE ./
  latency_flow_tests
Warning: Command line arguments for program should be given
after the program name. Assuming that plm_rsh_agent is a
command line argument for the program.
Warning: Command line arguments for program should be given
after the program name. Assuming that oarsh is a
command line argument for the program.
```

```

Unrecognized argument --mca ignored.
The authenticity of host 'capricorne-6.lyon.grid5000.fr (10.69.0.6)' can't
  be established.
RSA key fingerprint is d5:35:1a:e6:4b:12:25:c3:3d:2e:59:03:5c:fb:c2:58.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'capricorne-6.lyon.grid5000.fr,10.69.0.6' (RSA)
  to the list of known hosts.
Password:

```

Toujours sur Lyon, impossible de compiler avec `mpicc.openmpi` : ironie du sort, OpenMPI ne connaît pas MPI!

```

sbadia@capricorne-6:~/tests_debit_latence$ mpicc.openmpi
  latency_flow_tests.c -o latency_flow_tests
latency_flow_tests.c:22:17: error: mpi.h: No such file or directory
In file included from latency_flow_tests.c:24:
latency_flow_tests.h:28: error: expected '=', ',', ';', 'asm' or
  __attribute__ before 'BenchType'
latency_flow_tests.h:29: error: expected '=', ',', ';', 'asm' or
  __attribute__ before 'benchTypeTypes'
latency_flow_tests.h:30: error: expected '=', ',', ';', 'asm' or
  __attribute__ before 'testTypeTypes'
latency_flow_tests.h:31: error: expected '=', ',', ';', 'asm' or
  __attribute__ before 'benchTypeDisp'
latency_flow_tests.h:32: error: expected '=', ',', ';', 'asm' or
  __attribute__ before 'status'
latency_flow_tests.c: In function 'main':
latency_flow_tests.c:60: error: '\gls{MPI}_COMM_WORLD' undeclared (first
  use in this function)
latency_flow_tests.c:60: error: (Each undeclared identifier is reported
  only once
latency_flow_tests.c:60: error: for each function it appears in.)
latency_flow_tests.c: In function 'createBenchType':
latency_flow_tests.c:418: error: '\gls{MPI}_INT' undeclared (first use in
  this function)
[...]

```

Sur Nancy : l'utilisation de MPICH par défaut demande de lancer un programme en tâche de fond. Avec la version OpenMPI (`mpirun.openmpi`), il est impossible de faire communiquer les nœuds entre eux (comme Lyon, invitations de mot de passe).

Avec l'option `-mca` qui permet de forcer l'utilisation d'un autre protocole d'authentification, l'erreur suivante est rencontrée :

```

p1_15254: p4_error: interrupt SIGSEGV: 11
rm_l_1_15264: (0.589844) net_send: could not write to fd=5, errno = 32
p2_14980: p4_error: Found a dead connection while looking for messages: 0
rm_l_2_14990: (0.300781) net_send: could not write to fd=7, errno = 32
p3_7526: p4_error: net_recv read: probable EOF on socket: 1
rm_l_3_7536: (0.011719) net_send: could not write to fd=5, errno = 32

```

Déploiements

Les déploiements ne nous ont pas apportés entière satisfaction, ceux-ci auraient pu lisser complètement les problèmes précédents (compilation unique, car même environnement). Mais cela s'est passé autrement. Nous avons souvent rencontré des erreurs assez aléatoires, avec l'outil `kadeploy3`. Cet outil permet en effet de déployer son archive sur les nœuds

passés en paramètres, mais l'étendue de ce logiciel est énorme, puisqu'il s'occupe de faire démarrer les machines, les lancer depuis le réseau, vérifier les archives, les déployer, etc.

Il n'est pas rare de se retrouver avec le fameux message *Node not correctly deploy*, et de recommencer le déploiement pour que cela fonctionne finalement.

Des erreurs ont cependant été résolues, avec l'ajout des contrôleurs propriétaires (*sic*), mais la réussite du déploiement d'une image vers un autre site reste tout de même aléatoire.

Nous avons aussi utilisé l'outil `kaconsole`, qui permet de prendre le contrôle de la carte [Intelligent Platform Management Interface \(iPMI\)](#). Cette carte est un micro-ordinateur, qui permet de prendre le contrôle total de l'hôte, et donc d'accéder à l'hôte dans le cas où celui-ci ne répondrait plus en *SSH*. Cette interface permet aussi de voir passer le *BIOS*, ce qui est très pratique à distance.

Chapitre 5

Conclusion

*L'informatique n'est pas une science exacte,
on n'est jamais à l'abri d'un succès.*

1 Répartition du travail

Notre travail s'est divisé en deux équipes, chacune se concentrant sur un objectif particulier : Julien et Sébastien se sont chargés de la mise en place des outils pour l'utilisation des réseaux rapides ainsi que des **benchmarks** tandis que Guillaume et Luc se sont occupés de l'aspect inventaire du projet.

La formation à l'environnement a été consommatrice de temps et nous a occupé pendant un peu plus de deux semaines. Nous avons ensuite créé les équipes. L'équipe **benchmarks** a ensuite continué sa formation sur tous les aspects déploiement, réseaux rapides et **MPI**. Chacun a travaillé sur sa partie, de façon équitable et équilibrée.

Nous avons utilisé un système de version (*SVN*) installé sur un de nos serveurs pour partager nos travaux. Un *wiki* (développé dans le cadre de notre formation de cette année) a également été déployé pour partager nos connaissances, acquises sur l'environnement de Grid5000 et favoriser l'entraide au sein de notre groupe.

2 Problèmes généraux rencontrés

Le réseau de l'IUT ne nous permettant pas de sortir en *SSH* pour atteindre les infrastructures de Grid5000, nous avons dû utiliser le serveur de Julien comme passerelle en sortant sur le *port* 443 avec l'aide de l'outil **corkscrew**.

Il est heureux que le serveur de Julien était déjà configuré pour cette tâche : nous aurions sinon perdu un temps précieux.

Lors de l'utilisation de l'outil **kaconsole3**, le raccourci clavier pour en sortir est différent sur chaque et celui de Grenoble n'était pas documenté sur le *wiki* de Grid5000, occasionnant parfois des arrachages de cheveux (bug soumis aux administrateurs compétents, et résolu).

Le nombre important de maintenances sur les **clusters** nous a parfois empêché de travailler tel que nous l'avions prévu.

3 Résultats obtenus

Au terme de ce projet, nous constatons que nous avons réussi à satisfaire les différents objectifs qui nous avaient été imposés.

Ainsi, le projet Grid5000 dispose à présent d'outils leur permettant de vérifier automatiquement si le matériel livré est conforme à ce que la description impose. Un administrateur pourra alors compléter l'*OAR database* avec ces informations et lancer le script qui détectera les machines qui ne la respecte pas. Ce qu'on appelle la *recette* pour alors être validée - ou non - en temps et en heure. Avec les tests de **benchmarks** de disques durs, ceux-ci pourront particulièrement être vérifiés.

Afin de faire le bilan de la bonne interconnexion de tous les nœuds d'un même **cluster**, le projet dispose à présent d'un programme qui permettra de dresser une matrice complète des latences et débits constatés entre chacun des nœuds. Il sera également maintenant possible d'exporter cette matrice en *YAML* afin d'exploiter les résultats dans d'autres applications. Ou bien, directement utiliser un script que nous leur mettons à disposition pour exporter une page HTML colorée (mise en valeur des extrémités et des quartiles remarquables). Il leur sera aussi possible de tester la charge du réseau en réalisant des bisections au sein du **cluster** par le biais d'options et d'un second script. A partir des résultats, il sera enfin possible de tracer un graphique de la charge du réseau en fonction du nombre de nœuds, grâce à un dernier script, destiné à **gnuplot**.

Au niveau des réseaux rapides, les paquets Debian construits, ainsi que le tutoriels disponibles sous format \LaTeX et PDF, permettront de grandement faciliter la création d'environnements personnalisés qui supportent ces technologies.

Durant notre projet, nous avons également contribué au projet en reportant plusieurs *bugs* dans l'interface prévue à cet effet, et reporté plusieurs problèmes par la liste email des utilisateurs du projet. Enfin, les erreurs de matériels détectées par le biais de notre script d'inventaire ont déjà permises de mettre en évidences de nombreux problèmes sur les différents sites. Un administrateur de Grenoble (Guillaume Ranquet) s'est également intéressé à ce script pour l'utiliser lors d'un événement annuel de Grid5000, et nous a demandé de le rencontrer par visio-conférence au Loria (siège Nancéen de Grid5000).

4 Bénéfices personnels

Pour notre part, ce projet tuteuré aura été une occasion unique de découvrir le monde des **grilles de calcul** de **clusters**. Ainsi, nous avons appris à nous familiariser avec son vocabulaire et son fonctionnement, à travers les nombreuses pages de *wiki* du site de Grid5000, les outils OAR, nos recherches personnelles, et évidemment les nombreuses discussions que nous avons entretenues avec notre tuteur de projet.

Nous avons eu le plaisir de grandement nous perfectionner en *C*, de mettre en pratique de façon concrète les cours de *Ruby* enseignés cette année et enfin un perfectionnement de nos connaissances en *Perl*. *MPI* aura également été une première dans notre expérience de programmeur, puisque c'est une technologie qui s'apparente directement à la programmation pour les *Systèmes Multi-Agents (SMA)* ou les réseaux *Peer-to-Peer (P2P)*. Nous avons également eu ainsi l'occasion de découvrir les réseaux rapides. Nous avons aussi amélioré nos connaissances dans certains outils *Unix* nécessaires à la détection des matériels ou aux tests des disques durs.

D'un aspect plus humain, ce projet nous aura permis de nous apprendre à gérer une organisation impliquant quatre personnes sur un projet relativement long et complexe. Nous ne comptons pas les soirées passées à faire avancer le projet ou mieux nous former à l'environnement, ainsi que les nombreuses fois où le projet nous a permis de nous retrouver. Nous tenons également à souligner la présence de notre tuteur, Lucas Nussbaum, qui a toujours été prompt à nous répondre - parfois très tard - sur la messagerie *Jabber*, par email, à travers la liste des utilisateurs de Grid5000, ou simplement autour d'une bière.

Chapitre 6

Bibliographie

Toute tentative ratée de mettre en évidence la loi de Murphy est une mise en évidence de la loi de Murphy

Bibliographie

- [1] Groupe de réflexion. Grid'5000 plate-forme de recherche expérimentale en informatique. Technical report, Inria, Juillet 2003.
- [2] Isabelle BELIN. Le calcul haute performance est une technologie clé pour l'europe. *La recherche*, Juillet - Août 2009.
- [3] Douglas EADLINE. *High Performance Computing for Dummies*. Wiley Publishing, Inc, sun and amd special edition edition, 2009.
- [4] Michel SERRES. La simulation, technique nouvelle, ancienne tradition. *La recherche*, Janvier 2006.

Documents électroniques

- [5] <http://perldoc.perl.org>. Documentation officielle du langage *Perl*.
- [6] <http://rubybrain.com/>. Une API *Ruby* en Javascript.
- [7] <http://search.cpan.org/>. Le site de recherche de modules *Perl* du *CPAN*.
- [8] <http://www.grid5000.fr>. Site officiel du projet Grid5000.

Appendices

1 Script de vérification automatique du matériel

```
1 #!/usr/bin/perl
2 #This program is free software: you can redistribute it and/or modify
3 #it under the terms of the GNU General Public License as published by
4 #the Free Software Foundation, either version 3 of the License, or
5 #(at your option) any later version.
6 #
7 #This program is distributed in the hope that it will be useful,
8 #but WITHOUT ANY WARRANTY; without even the implied warranty of
9 #MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
10 #GNU General Public License for more details.
11 #
12 #You should have received a copy of the GNU General Public License
13 #along with this program. If not, see <http://www.gnu.org/licenses/>.
14
15 use warnings;
16 use strict;
17 use lib "lib/lib/perl5";
18 use YAML;
19 use XML::Simple;
20 use Data::Dumper;
21 use Math::Round;
22 use Getopt::Long;
23 use POSIX ":sys_wait_h";
24
25 my @criteria;
26 my $output;
27 my $help;
28 my $cow;
29 my $install;
30 my $remove;
31 my $liste_noeuds = $ENV{OAR_FILE_NODES} if (exists $ENV{OAR_FILE_NODES});
32 # Properties not in the OAR database
33 my @exclude = qw(disksize diskmodel);
34 # Properties which have no chance to be identical in OAR database
35 my @diff = qw(cputype ib10gmodel ib20gmodel myri10gmodel myri2gmodel);
36 my $error;
37 ## Default list of criteria
38 my @correct_crit = qw(cluster cpuarch cpucore cpufreq cputype diskmodel disksize
39                       disktype ib10g ib10gmodel ib20g ib20gmodel
40                       memnode myri10g myri10gmodel myri2g myri2gmodel
41                       );
42
43 Getopt::Long::GetOptions('o|output=s' => \$output,
44                          'f|file=s' => \$liste_noeuds,
45                          'p|parameter=s{,}' => \@criteria,
46                          'h|help' => \$help,
47                          'c|cow' => \$cow,
48                          'i|install' => \$install,
49                          'r|remove' => \$remove,
50                          );
51
52 ## Determining the list of the criteria
53 if ($#criteria < 0) {
54     # If no criteria specified, use the default list
55     @criteria = @correct_crit;
56 }
57 else {
58     # Check if the specified criteria are corrects
59     my $is_avail = 0;
60     foreach my $foo (@criteria) {
61         if (!(grep $_ eq $foo, @correct_crit)) {
62             print "One of the criteria you specified is not in the list of correct
63                   criteria.\n",
64                   "Please check your syntax and restart the program.\n";
65             for (@correct_crit) {
66                 print "$_ \n";
67             }
68             exit 1;
69         }
70     }
71 }
```

```

70 }
71
72 sub usage {
73     print "Ivcmp - copyright (c) 2010 Guillaume Delourmel <delourmel_guillaume@yahoo
74         .fr>, Luc Didry <lucdidry@free.fr>\n\n",
75     "Inventory script which compares the informations in the Oar database \n",
76     "and informations returned by Unix tools such as lshw, lspci, etc.\n",
77     "It can be used to see if all the nodes of a cluster have the same material\n\n",
78     "You need a ssh key with an empty passphrase in order to avoid to type the
79     passphrase for each node\n",
80     "You need to be root on the nodes in order to get all the required informations
81     with lshw\n",
82     "\nUsage : ./ivcmp [OPTIONS]...\n\n",
83     "  -f, --file [FILE]                specify the file containing the
84     list of the nodes (\$OAR_FILE_NODES by default)\n",
85     "                                  This file need to have one node
86     per line (duplicate nodes will be ignored)\n",
87     "  -o, --output [FILE]              print the result in the file in
88     YAML format\n",
89     "  -p, --parameter CRITERION [CRITERION]... check the nodes against specified
90     criteria\n",
91     "  -i, --install                    automatically install lshw on
92     nodes (lshw is required)\n",
93     "  -r, --remove                    automatically remove the
94     directories containing the ouput of the different used tools\n",
95     "  -h, --help                      print this help and exit\n\n",
96     "List of the correct criteria :\n";
97     for (@correct_crit) {
98         print "  $_\n";
99     }
100 }
101
102 ## Recover the node's properties in a hash table (OAR database)
103 sub RecupData {
104     my ($noeud, $sql) = @_ ;
105     my $file = "$noeud/oar_$noeud.yml";
106
107     # Recover the properties in YAML format
108     system "oarnodes -Y --sql $sql > $file";
109
110     open (FILE, "<", $file) or die("Unable to open file $file");
111
112     my $i = 1;
113     my @numb;
114     # Get the different id of the node in the OAR database
115     while (<FILE>) {
116         my $foo = $_;
117         chomp $foo;
118         if ($foo =~ /\^d*:\s?$/ ) {
119             push @numb, split(":", $foo);
120         }
121     }
122     close(FILE);
123
124     my $yaml = YAML::LoadFile($file);
125
126     ## Here are the properties we need to check in the OAR database
127     my @listValeur = qw(network_address cluster cpuarch cpucore cputype
128         disktype ethnb ib10g ib10gmodel ib20g ib20gmodel memnode myri10g
129         myri10gmodel myri2g myri2gmodel);
130
131     my %tab_oar;
132     for (@listValeur) {
133         my $temp = Dump($yaml->{$numb[0]}->{$_});
134         $temp =~ s/\^---- //;
135         chomp $temp;
136         # Unification of format
137         $temp =~ s/\^s*YES|TRUE\s*$/1/i;
138         $temp =~ s/\^s*NO|FALSE|NONE|\^s*$/~/i;
139         $tab_oar{$_} = $temp;
140     }
141     # Check if all the cpufreq are the same

```

```

134 my $first = 0;
135 for (@numb) {
136     if (!$first) {
137         my $stemp = Dump($yaml->{$_}->{'cpufreq'});
138         $stemp =~ s/\s|--- //g;
139         chomp $stemp;
140         $stab_oar{'cpufreq'} = nearest(0.1, $stemp);
141     }
142     else {
143         my $stemp = Dump($yaml->{$_}->{'cpufreq'});
144         $stemp =~ s/\s|--- //g;
145         chomp $stemp;
146         my $scpufreq = nearest(0.1, $stemp);
147         if ($stab_oar{'cpufreq'} != $scpufreq) {
148             $stab_oar{"cpufreq_.$first"} = $scpufreq;
149             $error .= "----- The node $noeud has differents frequency on its cores
                    according to the Oar Database -----\n";
150         }
151     }
152     $first++;
153 }
154 return %tab_oar;
155 }
156
157 ## Format lshw file
158 sub etete_lshw {
159     my $done = 0;
160     my $start = "<node";
161     my ($file) = @_;
162     my $file2 = $file.".tmp";
163     open(FICHER, "<", $file) or die("Unable to open file $file");
164     open(FICHIERTMP, ">", $file2) or die("Unable to open file $file2");
165
166     my $disk = 0;
167     my $disk2 = 0;
168     while(defined(my $tmp = <FICHER>)) {
169         chomp($tmp);
170         if (!$done && "$tmp" =~ /\s*$start/) {
171             $done = 1;
172             print FICHIERTMP "<perldata>\n";
173         }
174         ## If we have two disks nodes
175         if ($disk && "$tmp" =~ /\s*node id="disk".*$/) {
176             $disk2 = 1;
177         }
178         if (!$disk && "$tmp" =~ /\s*node id="disk".*$/) {
179             $disk = 1;
180         }
181         if (!$disk2) {
182             print FICHIERTMP $tmp."\n" if ($done);
183         }
184         if ($disk2 && "$tmp" =~ /\s*\node>$/) {
185             $disk2 = 0;
186         }
187     }
188     print FICHIERTMP "</perldata>\n";
189     close(FICHER);
190     close(FICHIERTMP);
191     rename $file2, $file;
192     return $file;
193 }
194
195 ## Get properties with Unix tools and put them in files
196 sub CreateFiles {
197     my ($noeud) = @_;
198     ## uname
199     system "ssh root@$noeud -o \"StrictHostKeyChecking no\" uname -nm > $noeud/
        uname_$noeud.txt";
200     ## lspci
201     system "ssh root@$noeud lspci > $noeud/lspci_$noeud.txt";
202     ## cat /proc/cpuinfo
203     system "ssh root@$noeud cat /proc/cpuinfo > $noeud/cpuinfo_$noeud.txt";
204     ## lshw

```

```

205     system "ssh root@$noeud lshw -xml -C disk -C memory -C network -C processor >
        $noeud/lshw_$noeud.xml";
206     ## ifconfig
207     system "ssh root@$noeud ifconfig > $noeud/ifconfig_$noeud.txt";
208 }
209
210 ## Gathers informations of Unix tools
211 sub ParseFiles {
212     my ($noeud) = @_;
213     my %properties;
214     my @listValeur = qw(network_address cluster cpuarch cpucore cpufreq cputype
215     disktype ethnb ib10g ib10gmodel ib20g ib20gmodel memnode myri10g
216     myri10gmodel myri2g myri2gmodel disksize);
217     # Format the lshw file
218     my $tmp_xml_lshw = etete_lshw("$noeud/lshw_$noeud.xml");
219     # We define "id" as a key for the element
220     my $xml = new XML::Simple (KeyAttr=>'id');
221     my $oar = $xml->XMLin($tmp_xml_lshw);
222
223     $properties{cputype} = $oar->{node}->{"cpu:0"}->{product};
224     $properties{cputype} =~ s/(\s+)/ /g;
225     #####
226     $properties{memnode} = $oar->{node}->{"memory"}->{size}->{content};
227     if (!defined($properties{memnode})) {
228         my $i = 0;
229         while (defined($oar->{node}->{"memory:$i"})) {
230             my $j = 0;
231             while (defined($oar->{node}->{"memory:$i"}->{node}->{"bank:$j"})) {
232                 if (my $stemp2 = $oar->{node}->{"memory:$i"}->{node}->{"bank:$j"}->{size}->{
                content}) {
233                     $properties{memnode} += $stemp2;
234                 }
235                 $j++;
236             }
237             $i++;
238         }
239     }
240     $properties{memnode} = nearest(512, $properties{memnode}/1024/1024);
241     #####
242     $properties{disksize} = $oar->{node}->{"disk"}->{size}->{content};
243     if (!defined($properties{disksize})) {
244         my $i = 0;
245         while (exists $oar->{node}->{"disk:$i"}) {
246             if (!defined($properties{diskmodel})) {
247                 $properties{diskmodel} = $oar->{node}->{"disk:$i"}->{product};
248             #         if (!defined($properties{diskmodel})) {
249             #             $properties{diskmodel} = $oar->{node}->{"disk:$i"}->{description};
250             #         }
251             }
252             if (defined($oar->{node}->{"disk:$i"}->{size}->{content})) {
253                 $properties{disksize} += $oar->{node}->{"disk:$i"}->{size}->{content};
254             }
255             $i++;
256         }
257     }
258     else {
259         $properties{diskmodel} = $oar->{node}->{"disk"}->{product};
260         if (!defined($properties{diskmodel})) {
261             $properties{diskmodel} = $oar->{node}->{"disk"}->{description};
262         }
263     }
264     chomp $properties{disksize};
265     $properties{disksize} = round($properties{disksize}/1000000000);
266     #####
267     my $i = 0;
268     while (exists $oar->{node}->{"cpu:$i"}) {
269         my $scpu = $oar->{node}->{"cpu:$i"}->{size}->{content};
270         chomp $scpu;
271         $scpu = $scpu/1000000000;
272         $scpu = nearest(.1, $scpu);
273         if (!$i) {
274             $properties{cpufreq} = $scpu;
275         }

```

```

276     else {
277         if ($cpu != $properties{cpufreq}) {
278             $properties{"cpufreq_$i"} = $cpu;
279             $error .= "----- The node $noeud has different frequency on its cores
                according to the received data -----\n";
280         }
281     }
282     $i++;
283 }
284 ## uname
285 open(UNAME, "<", "$noeud/uname_$noeud.txt") or die ("Unable to open file $noeud/
    uname_$noeud.txt : $!\n");
286 while (defined(my $foo = <UNAME>)) {
287     chomp $foo;
288     ($properties{network_address}, $properties{cpuarch}) = split(" ", $foo);
289     $properties{cluster} = $1 if ($properties{network_address} =~ /^([\^-\]*).*$/);
290 }
291 close(UNAME);
292
293 ## cpuinfo
294 open(CPUINFO, "<", "$noeud/cpuinfo_$noeud.txt") or die ("Unable to open file
    $noeud/cpuinfo_$noeud.txt : $!\n");
295 while (defined(my $foo = <CPUINFO>)) {
296     chomp $foo;
297     $properties{cpucore} = $1 if ($foo =~ /core\sid\s*:\s([0-9])/i);
298 }
299 close(CPUINFO);
300 $properties{cpucore}++;
301
302 ## lspci
303 open(LSPCI, "<", "$noeud/lspci_$noeud.txt") or die ("Unable to open file $noeud/
    lspci_$noeud.txt : $!\n");
304 while (defined(my $foo = <LSPCI>)) {
305     chomp $foo;
306     if ($foo =~ /(MT23108|MT25208|MT25408|MNEH18-XTC|MT25418)/i) {
307         $properties{ib10g} = 1;
308         $properties{ib10gmodel} = $1;
309     }
310     if ($foo =~ /(MHGH29-XTC|MT26418)/i) {
311         $properties{ib20g} = 1;
312         $properties{ib20gmodel} = $1;
313     }
314     if ($foo =~ /(10G-PCIE-8A)/i) {
315         $properties{myri10g} = 1;
316         $properties{myri10gmodel} = $1;
317     }
318     if ($foo =~ /(M3F-PCIXD-2|M3F-PCIXF-2|Myrinet 2000)/i) {
319         $properties{myri2g} = 1;
320         if ($1 =~ /(M3F-PCIXD-2|M3F-PCIXF-2)/i) {
321             $properties{myri2gmodel} = $1;
322         }
323         else {
324             $error .= "----- The Unix tools don't say the Myri2G model. Please look at the
                OAR Database -----\n";
325         }
326     }
327 }
328 $properties{disktype} = `egrep -i -m 1 "sas|sata|scsi" $noeud/lspci_$noeud.txt
    `;
329 if (!( $properties{disktype} =~ s/^.*(sas|sata|scsi).*$/\1/i)) {
330     $properties{disktype} = `egrep -i -m 1 "ide" $noeud/lspci_$noeud.txt `;
331     $properties{disktype} =~ s/^.*(ide).*$/\1/i;
332 }
333 chomp($properties{disktype});
334 $properties{disktype} = lc($properties{disktype});
335
336 ## ifconfig
337 my $inet = `grep -c "inet addr" $noeud/ifconfig_$noeud.txt `;
338 my $lo = `grep -c -e "lo[ ]" $noeud/ifconfig_$noeud.txt `;
339 chomp $inet;
340 chomp $lo;
341 $properties{ethnb} = $inet - $lo;
342 }

```

```

343 close(LSPCI);
344
345 for (@listValeur) {
346     $properties{$_} = "" if (!$properties{$_});
347     chomp $properties{$_};
348 }
349 return %properties;
350 }
351
352 #####
353 # Beginning of the main program
354 #####
355 if ($help)
356 {
357     &usage;
358     exit 0;
359 }
360
361 my $answer = "42";
362 ## Check if the output file already exists
363 if ($output) {
364     if (-e $output) {
365         while ($answer !~ /^(y|n)/i) {
366             print "The output file already exists. Do you want to overwrite it ? (Y/N) "
367                 ;
368             $answer = <STDIN>;
369             chomp $answer;
370             if ($answer eq "") {
371                 $answer = "Y";
372             }
373             elsif ($answer !~ /^y|n/i) {
374                 print "You need to answer y or n !\n";
375             }
376         }
377     }
378     if (lc($answer) eq "n") {
379         print "You don't want to overwrite the ouput file : aborting the program\n";
380         exit 0;
381     }
382 }
383 ## Recovery of the reserved nodes list
384 open (LISTE_NOEUDS, "<", $liste_noeuds) or die("Unable to open file $liste_noeuds :
385     $!\n");
386
387 ## Get a hash of nodes' name and create sql request parameters
388 my %liste_noeuds;
389 while (defined(my $foo = <LISTE_NOEUDS>)) {
390     chomp $foo;
391     $liste_noeuds{$foo} = "\"network_address='$foo'\"";
392 }
393 close(LISTE_NOEUDS);
394
395 ## Get a hash of nodes' name with a hash of their properties
396 my @liste_noeuds = keys %liste_noeuds;
397 # Hash of the properties returned by the OAR Database
398 my %properties_oar;
399 # Hash of the properties returned by the Unix tools
400 my %properties_parse;
401 # Automatic install of lshw on deployed nodes
402 my $pid;
403 for (@liste_noeuds) {
404     my $foo = $_;
405     $pid = fork();
406     # Children
407     if ($pid == 0) {
408         my $ping = system "ping $foo -c 1 &>/dev/null";
409         $ping >>= 8;
410         if ($ping) {
411             delete $liste_noeuds{$foo};
412         }
413     }
414     else {
415         if ($install) {

```

```

414     'ssh root\@$foo -o \"StrictHostKeyChecking no\" \"apt-get update &>/dev/null &&
      apt-get install lshw -y &>/dev/null\"'
415     }
416     unless (-d $foo && -e $foo) {
417     mkdir ($foo) or die ("Unable to create directory $foo : !\n");
418     }
419     &CreateFiles($foo);
420     }
421     exit(0);
422 }
423 }
424 # Parent
425 if ($pid != 0) {
426     do {
427         $pid = waitpid(-1,&WNOHANG);
428     }
429     until ($pid == -1);
430 }
431
432 for (@liste_noeuds) {
433     my $foo = $_;
434     my $ping = system "ping $foo -c 1 &>/dev/null";
435     $ping >>= 8;
436     if ($ping) {
437         delete $liste_noeuds{$foo};
438     }
439     else {
440         ## Get Oar database infos and put them in some files
441         my %temp = &RecupData($foo, $liste_noeuds{$foo});
442         $properties_oar{$foo} = \%temp;
443
444         ## Get node infos (Unix tools)
445         my %temp2 = &ParseFiles($foo);
446         $properties_parse{$foo} = \%temp2;
447
448         ## Compare oar's properties and nodes' properties
449         my $oar = $properties_oar{$foo};
450         my $parse = $properties_parse{$foo};
451         for (keys %$parse) {
452             my $titi = $_;
453             my $exclude = 0;
454             my $diff = 0;
455             for (@exclude) {
456                 $exclude = 1 if ("$_" eq "$titi");
457             }
458             for (@diff) {
459                 $diff = 1 if ("$_" eq "$titi");
460             }
461             if (!$exclude && exists $oar->{$titi} && $parse->{$titi} ne $oar->{$titi}) {
462                 # Add an error message if there is a difference and this is not predictable
463                 $error .= "----- The Oar database $titi information of the node $foo is not the
464                     same as we check on the node -----\n".
465                     "         OAR database : $oar->{$titi}           Unix tools : $parse->{$titi}
466                     }\n" if !$diff;
467             }
468         }
469     }
470 }
471
472 ## Check if the nodes are identical, based on the criteria list
473 # Here we use the Unix tools list since it's more complete
474 # and more reliable
475 my $first = "42";
476 my %resultat;
477 #delete $properties_oar{"");
478 for (keys %properties_oar) {
479     my $foo = $_;
480     my $parse = $properties_parse{$foo};
481     # Firt node : first group
482     if ($first) {
483         $resultat{"1"}->{"nodes"}->{$foo} = "~";
484         foreach my $bar (@criteria) {
485             $resultat{"1"}->{"$bar"} = $parse->{$bar}

```

```

484     }
485     $first = 0;
486 }
487 else {
488     my $groupe;
489     my @temp = keys %resultat;
490     # Compare to all groups
491     for (@temp) {
492         my $key = $_;
493         my $ok = "42";
494         for (@criteria) {
495             my $bar = $_;
496             if ($resultat{"$key"}->{"$bar"} ne $parse->{"$bar"}) {
497                 $ok = 0;
498                 last;
499             }
500         }
501         # If all criteria are ok, we're in the good group
502         if ($ok){
503             $groupe = $key;
504             last;
505         }
506         else {
507             $groupe = 0;
508         }
509     }
510     # Creation of a new group if we need
511     if (!$groupe) {
512         $groupe = $#temp + 2;
513         for (@criteria) {
514             my $bar = $_;
515             $resultat{"$groupe"}->{"$bar"} = $parse->{"$bar"};
516         }
517     }
518     # Put the node in its group
519     $resultat{"$groupe"}->{"nodes"}->{"$foo"} = "~";
520 }
521 }
522
523 if ($cow) {
524     print "
525         \n",
526         "\/
527         \\ \n";
528 }
529 print "—— Notice that the printed value are from Unix tools , not from OAR database
530 ——\n";
531 ## Prepare the results to be print
532 my $out;
533 for (sort keys %resultat) {
534     # group
535     $out .= "$_ :\n";
536     my $bar = $resultat{$_};
537     for (@criteria) {
538         $out .= "  $_ : $bar->{$_}\n";
539     }
540     $out .= "  nodes :\n";
541     my $nodes = $bar->{"nodes"};
542     for (sort keys %$nodes) {
543         $out .= "    $_ : $nodes->{$_}\n";
544     }
545 }
546 ## Print results (in YAML format)
547 if (!$output) {
548     print $out;
549 }
550 else {
551     # Print results (in YAML format) in a specified file
552     open (OUT, ">", $output) or die("Unable to open file $output");
553     print OUT $out;
554     if ($cow) {

```

```

552     print "|                Bazinga !                |\n";
553 }
554 close(OUT);
555 }
556
557 ## Print errors (properties differences between OAR and Unix tools)
558 print $error unless !defined($error);
559 if ($cow) {
560     print "\\
561         /\n",
562         "      /\n",
563         "      (oo)\n",
564         "      (oo)\n",
565         "      ||----w |\n",
566         "      ||      |\n";
567 }
568 ## Remove the nodes' directories
569 if ($remove) {
570     for (keys %liste_noeuds) {
571         'rm -rf $_';
572     }
573 }

```

2 README du script ivcmp

```

1 #####
2 #####
3
4 WHAT IS IVCMP ?
5
6 Ivcmp is a Perl script which gets hardware informations from the reserved nodes,
7 compares them to the OAR Database and determines if all nodes have the same
8 hardware.
9 It uses lshw, ifconfig, lspci, uname and /proc/cpuinfo to get the informations
10 from the nodes.
11 The output shows you the nodes sorted in groups made by nodes which have
12 exactly the same material, in YAML format. The hardware informations used to
13 compare the nodes are :
14     cluster      : name of the cluster the node belongs to
15     cpuarch      : architecture (x86_64 by ex.)
16     cpucore      : number of cores by CPU
17     cpufreq      : frequency of the CPUs
18     cputype      : type of CPU
19     diskmodel    : model of hard disk drive
20     disksize     : size of hard disk drive
21     disktype     : hard disk drive interface
22     ib10g        : do the node has an Infiniband 10g card ?
23     ib10gmodel   : which model ?
24     ib20g        : do the node has an Infiniband 20g card ?
25     ib20gmodel   : which model ?
26     memnode      : ram quantity
27     myri10g      : do the node has a Myrinet 10g card ?
28     myri10gmodel : which model ?
29     myri2g       : do the node has a Myrinet 2g card ?
30     myri2gmodel  : which model ?
31
32 The differences with the OAR Database will be displayed at the end of the YAML
33 output.
34 You can specify a file where to store the YAML output.
35
36 #####
37 #####
38
39 HOW TO USE IT ?
40
41 You need a ssh key with an empty passphrase in order to avoid to type the
42 passphrase for each node.
43 You need to be root on the nodes in order to get all the required informations
44 with lshw.
45

```

```

46 Usage : ./ivcmp [OPTIONS]...
47
48 -f, --file [FILE]
49     specify the file containing the list of the nodes ($OAR_FILE_NODES by
50     default) This file need to have one node per line (duplicate nodes will be
51     ignored)
52 -o, --output [FILE]
53     print the result in the file in YAML format
54 -p, --parameter CRITERION [CRITERION]...
55     check the nodes against specified criteria
56 -i, --install
57     automatically install lshw on nodes
58 -r, --remove
59     automatically remove the directories containing the ouput of the different
60     used tools
61 -h, --help
62     print help and exit
63
64 List of the correct criteria : see the list above
65
66 Example of inconsistency warning :
67
68 ----- The Oar database disktype information of the node -----
69     capricorne-54.lyon.grid5000.fr is not the same as we check on the node -----
70     OAR database : scsi           Unix tools : sata
71
72 Example of YAML output :
73
74 1 :
75   cluster : sagittaire
76   cpuarch : x86_64
77   cpucore : 1
78   cpufreq : 2.4
79   cputype : AMD Opteron(tm) Processor 250
80   diskmodel : MAT3073NC
81   disksize : 74
82   disktype : scsi
83   ib10g : ~
84   ib10gmodel : ~
85   ib20g : ~
86   ib20gmodel : ~
87   memnode : 2048
88   myri10g : ~
89   myri10gmodel : ~
90   myri2g : ~
91   myri2gmodel : ~
92   nodes :
93     sagittaire-39.lyon.grid5000.fr : ~
94     sagittaire-42.lyon.grid5000.fr : ~
95     sagittaire-44.lyon.grid5000.fr : ~
96     sagittaire-45.lyon.grid5000.fr : ~
97     sagittaire-46.lyon.grid5000.fr : ~
98     sagittaire-48.lyon.grid5000.fr : ~
99     sagittaire-52.lyon.grid5000.fr : ~
100    sagittaire-53.lyon.grid5000.fr : ~
101    sagittaire-54.lyon.grid5000.fr : ~
102    sagittaire-61.lyon.grid5000.fr : ~
103    sagittaire-63.lyon.grid5000.fr : ~
104    sagittaire-66.lyon.grid5000.fr : ~
105 2 :
106   cluster : capricorne
107   cpuarch : x86_64
108   cpucore : 1
109   cpufreq : 2
110   cputype : AMD Opteron(tm) Processor 246
111   diskmodel : SCSI Disk
112   disksize : 36
113   disktype : sata
114   ib10g : ~
115   ib10gmodel : ~
116   ib20g : ~
117   ib20gmodel : ~
118   memnode : 2048

```

```

119 myri10g : ~
120 myri10gmodel : ~
121 myri2g : 1
122 myri2gmodel : ~
123 nodes :
124   capricorne-33.lyon.grid5000.fr : ~
125   capricorne-34.lyon.grid5000.fr : ~
126   capricorne-35.lyon.grid5000.fr : ~
127   capricorne-36.lyon.grid5000.fr : ~
128   capricorne-37.lyon.grid5000.fr : ~
129   capricorne-38.lyon.grid5000.fr : ~
130   capricorne-39.lyon.grid5000.fr : ~
131   capricorne-40.lyon.grid5000.fr : ~
132   capricorne-41.lyon.grid5000.fr : ~

```

3 Script de benchmark des disques durs

```

1  #!/usr/bin/perl
2  #This program is free software: you can redistribute it and/or modify
3  #it under the terms of the GNU General Public License as published by
4  #the Free Software Foundation, either version 3 of the License, or
5  #(at your option) any later version.
6  #
7  #This program is distributed in the hope that it will be useful,
8  #but WITHOUT ANY WARRANTY; without even the implied warranty of
9  #MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
10 #GNU General Public License for more details.
11 #
12 #You should have received a copy of the GNU General Public License
13 #along with this program. If not, see <http://www.gnu.org/licenses/>.
14 use warnings;
15 use strict;
16 use lib "lib/lib/perl5";
17 use POSIX ":sys_wait_h";
18 use Getopt::Long;
19 use Math::Round;
20
21 my $install;
22 my $help;
23 my $output;
24 my $type;
25 my $filelist;
26 my $remove;
27 Getopt::Long::GetOptions('f|file=s' => \$filelist ,
28                          'h|help' => \$help ,
29                          'y|yaml' => \$type ,
30                          'r|remove' => \$remove ,
31                          'o|output=s' => \$output ,
32                          'i|install' => \$install ,
33                          );
34
35 sub usage {
36     print "Ivbon - copyright (c) 2010 Guillaume Delourmel <delourmel_guillaume@yahoo
37         .fr>\n\n",
38         "Script to test all the hard disk drive of the nodes on a cluster\n",
39         "Usage : ivbon [OPTIONS]...\n",
40         "  -f, --file [FILE]           file which contains the list of node to test\n",
41         "  -i, --install                automatically install bonnie++ on
42         nodes (bonnie++ is required)\n",
43         "  -y, --yaml                    output test as yaml\n",
44         "  -r, --remove                remove all temporary files create
45         by the script\n",
46         "  -o, --output [FILE]         print the result in the file in
47         YAML format, default is bonnie.txt\n",
48         "  -h, --help                  print this help and exit\n\n";
49 }
50
51 if ($help)
52 {
53     &usage;
54     exit 0;
55 }

```

```

51 }
52
53 if (!$output){
54     $output="bonnie.yml";
55 }
56
57 my $answer = "42";
58 ## Check if the output file already exists
59 if ($output) {
60     if (-e $output) {
61         while ($answer !~ /^(y|n)/i) {
62             print "The output file already exists. Do you want to overwrite it ? (Y\|n) "
63                 ;
64             $answer = <STDIN>;
65             chomp $answer;
66             if ($answer eq "") {
67                 $answer = "Y";
68             }
69             elsif ($answer !~ /^y|n/i) {
70                 print "You need to answer y or n !\n";
71             }
72         }
73     }
74     if (lc($answer) eq "n") {
75         print "You don't want to overwrite the ouput file : aborting the program\n";
76         exit 0;
77     }
78 }
79 my %liste_noeuds;
80 my $filelist;
81 if ($filelist){
82     # Recovery of the nodes list specified by User
83     open (LISTE_NOEUDS, "<" , $filelist) or die("Unable to open file $filelist : $!\n"
84         );
85     while (defined(my $foo = <LISTE_NOEUDS>)) {
86         chomp $foo;
87         $liste_noeuds{$foo} = "";
88     }
89     close(LISTE_NOEUDS);
90 }else{
91     # Recovery of the reserved nodes list
92     $liste_noeuds = $ENV{OAR_FILE_NODES} if (exists $ENV{OAR_FILE_NODES});
93     open (LISTE_NOEUDS, "<" , $liste_noeuds) or die("Unable to open file $liste_noeuds
94         : $!\n");
95     # Get a hash of nodes' name and create sql request parameters
96     while (defined(my $foo = <LISTE_NOEUDS>)) {
97         chomp $foo;
98         $liste_noeuds{$foo} = "";
99     }
100     close(LISTE_NOEUDS);
101 }
102
103 # Automatic install of Bonnie++ on deployed nodes
104 my @liste_noeuds = keys %liste_noeuds;
105
106 ## Launch the hard disk drive test in parallele on the deployed node
107 my $pid;
108 foreach (@liste_noeuds){
109     my $node=$_;
110     $pid = fork();
111     if ($pid == 0) {
112         if ($install){
113             print "Installation sur $node\n";
114             system("ssh root@$node -o \"StrictHostKeyChecking no\" \"apt-get update
115                 &&>/dev/null && apt-get install bonnie++ -y &>/dev/null\"");
116         }
117         system("ssh root@$node \"free -m | grep Mem\" > $node.mem");
118         my $ram;
119         open(MEMFILE,"<",$node.mem) or die("Unable to open file $node.mem");
120         while (defined(my $foo = <MEMFILE>)){
121             if ($foo =~ /^Mem:\s*(\w*)\s*/){
122                 $ram= $1;

```

```

120 }
121 }
122 $ram=$ram*2;
123 system("ssh root@$node \"bonnie++ -s $ram -d /tmp -m $node -u root\"> $node.
    txt");
124 exit;
125 }
126 }
127 }
128 if ($pid != 0) {
129 do
130 {
131 $pid = waitpid(-1,&WNOHANG);
132 }
133 until ($pid == -1);
134 }
135 my $txt="";
136
137 ##Build the output file
138 if($type){
139 my $moyenne_ecriture=0;
140 my $moyenne_lecture=0;
141 my $moyenne_nbop=0;
142 foreach (@liste_noeuds){
143 open(FICHER, "<", "$_.txt") or die("Unable to open file $_.txt");
144 while(defined(my $foo = <FICHER> )){
145 if($foo =~ /(.*\.grid5000.fr),\w*,\w*,\w*,(\w*),\w*,\w*,\w*,\w*,\w*,(\w*),\w
    *,(\w*.\w),.*/){
146 my $secr=$2;
147 my $lec=$3;
148 my $nbo=$4;
149 $txt="$txt$1\n";
150 $txt="$txt writing: $secr\n";
151 $txt="$txt reading: $lec\n";
152 $txt="$txt number_of_operations: $nbo\n";
153 $moyenne_ecriture = $moyenne_ecriture+$secr;
154 $moyenne_lecture = $moyenne_lecture+$lec;
155 $moyenne_nbop = $moyenne_nbop+$nbo;
156 }
157 }
158 }
159 $moyenne_ecriture = $moyenne_ecriture / @liste_noeuds;
160 $moyenne_lecture = $moyenne_lecture / @liste_noeuds;
161 $moyenne_nbop = $moyenne_nbop / @liste_noeuds;
162
163 $moyenne_ecriture = nearest(0.1,$moyenne_ecriture);
164 $moyenne_lecture = nearest(0.1,$moyenne_lecture);
165 $moyenne_nbop = nearest(0.1, $moyenne_nbop);
166 $txt="Average_number_of_operations: $moyenne_nbop\n$txt";
167 $txt="Average_speed_of_reading: $moyenne_lecture\n$txt";
168 $txt="Average_speed_of_writing: $moyenne_ecriture\n$txt";
169
170 }else{
171 my $moyenne_ecriture=0;
172 my $moyenne_lecture=0;
173 my $moyenne_nbop=0;
174
175 $txt="\t\tNode\t\t\t\t|Writing\t|Reading\t|Number_of_operations\n";
176 foreach (@liste_noeuds){
177 open(FICHER, "<", "$_.txt") or die("Unable to open file $_.txt");
178 while(defined(my $foo = <FICHER> )){
179 if($foo =~ /(.*\.grid5000.fr),\w*,\w*,\w*,(\w*),\w*,\w*,\w*,\w*,\w*,(\w*),\w
    *,(\w*.\w),.*/){
180 my $taille=0;
181 my $bar=$1;
182 my $secr=$2;
183 my $lec=$3;
184 my $nbo=$4;
185 $txt = "$txt$1\t";
186 $taille++ while ($bar =~ /(.)\/g);
187 if($taille < 24){
188 $txt = "$txt\t\t\t\t|";
189 }elseif($taille > 32){

```

```

190         $txt = "$txt|";
191     }else{
192         $txt = "$txt\t|";
193     }
194     $txt = "$txt$ecr\t\t|$lec\t\t|$nbo\n";
195     $moyenne_ecriture = $moyenne_ecriture+$ecr;
196     $moyenne_lecture = $moyenne_lecture+$lec;
197     $moyenne_nbop = $moyenne_nbop+$nbo;
198     }
199 }
200
201 }
202 $moyenne_ecriture = $moyenne_ecriture / @liste_noeuds;
203 $moyenne_lecture = $moyenne_lecture / @liste_noeuds;
204 $moyenne_nbop = $moyenne_nbop / @liste_noeuds;
205
206 $moyenne_ecriture = nearest(0.1,$moyenne_ecriture);
207 $moyenne_lecture = nearest(0.1,$moyenne_lecture);
208 $moyenne_nbop = nearest(0.1, $moyenne_nbop);
209
210 $txt="Average_number_of_operations: $moyenne_nbop\n$txt";
211 $txt="Average_speed_of_reading: $moyenne_lecture\n$txt";
212 $txt="Average_speed_of_writing: $moyenne_ecriture\n$txt";
213
214 }
215 open (FICHIERYAML,">",$output)or die("Unable to open file : $output");
216 print FICHIERYAML $txt;
217 close(FICHIERYAML);
218
219
220 if($remove){
221     foreach (@liste_noeuds){
222         'rm $_.mem $_.txt';
223     }
224 }

```

4 README du script ivbon

```

1 #####
2 #####
3
4 WHAT IS IVBON ?
5
6 Ivbon is a script to check the hard disk drive performance for the different
7 node on the cluster of Grid5000.
8 This is a perl script who use the Bonnie++ tools to perform the test , it
9 connects to the different nodes previously reserved and tests their hard disk
10 drive after the script summarizes the informations.
11 This may take a long time according to the memory ram use to make the test
12 because Bonnie++ take a file size which make the double of memory size to
13 perform the test.
14
15 #####
16 #####
17
18 HOW TO USE ?
19
20 The first time use the script with option -i to install Bonnie++ on the node,
21 then you can use the differents options like :
22 -y      :      to have a yaml output
23 -o file :      to precise the name of output file
24 -h      :      to print the help
25 -f file  :      specified the node name in a file
26 -r      :      remove the temporary file create by the script
27
28 At the end of the script you can have the result in Yaml (use -y) :
29
30     Average_speed_of_writing: 30862
31     Average_speed_of_reading: 66291.3
32     Average_number_of_operations: 205.3
33     grelon -33.nancy.grid5000.fr
34     writing: 31076

```

```

35     reading: 68539
36     Number_of_operations: 207.8
37     grelon -34.nancy.grid5000.fr
38     writing: 30361
39     reading: 63508
40     Number_of_operations: 208.7
41     grelon -35.nancy.grid5000.fr
42     writing: 31149
43     reading: 66827
44     Number_of_operations: 199.4
45
46 or in text :
47
48     Average_speed_of_writing: 30774.3
49     Average_speed_of_reading: 63275.7
50     Average_number_of_operations: 190.4
51     Node                               |writing      |Reading      |
52     grelon -33.nancy.grid5000.fr       |31268        |65192        |186.8
53     grelon -34.nancy.grid5000.fr       |29966        |60931        |190.2
54     grelon -35.nancy.grid5000.fr       |31089        |63704        |194.3

```

5 Benchmark de Latence et Débit

```

1  /*
2  * OAR LATENCY FLOW TESTS
3  *
4  * Cree une matrice des debits et latence entre chaque noeud (sauf le rank 0) de l'
5  * execution MPI, dans les deux sens,
6  * ou separe la liste des noeuds en deux pour faire une bisection.
7  *
8  * OPTIONS : Voir -h
9  *
10 * AUTEURS : <julien@vaubourg.com>
11 *           <seb@sebian.fr> <badia.seb@gmail.com>
12 * 2010, pour Grid5000
13 *
14 */
15
16 #include <stdio.h>
17 #include <stdlib.h>
18 #include <string.h>
19 #include <time.h>
20 #include <unistd.h>
21 #include <math.h>
22 #include <mpi.h>
23
24 #include "latency_flow_tests.h"
25
26 /* A compiler avec un compilateur MPI et a executer avec une commande de la meme
27 * categorie, sur une reservation de 3 noeuds ou plus */
28 int main(int argc, char** argv) {
29     int
30         rank, /* Numero du noeud qui execute le script, par rapport au nb de noeuds
31                concernees par l'execution */
32         nbNodes, /* Nombre de noeuds concernees par l'execution du programme */
33         pktSize, /* Taille du mot qui sera envoye pour faire les tests de debit */
34         nbRetry, /* Nb de fois qu'un test sera recommence afin d'augmenter la
35                precision */
36         bisection, /* Mode bisection ? */
37         randBiss, /* Mode bisection, avec formation des paires aleatoires ? */
38         yaml, /* Sortie dans un fichier YAML (yamlFile) en plus de la matrice ? */
39         gnuplot, /* Sortie pour un graphique gnuplot plutot qu'une matrice ? */
40         i, sender, recver, l; /* Divers compteurs */
41     float
42         sumLatency, /* Somme de toutes les latences d'un meme test afin de pouvoir
43                faire la moyenne */
44         sumFlow; /* Idem pour le debit */
45     YourTest
46         *bissTests, /* Tableau de tous les tests a envoyer, sert pour le rank 0 */
47         myTest; /* Test que recevra le noeud si il n'est pas le rank 0 */

```

```

44     Bench
45     *sameBenchs; /* Tableau qui recevra tous les resultats des benchs d'un meme
        test , a partir desquels on fera des moyennes */
46     MyResult
47     myResult , /* Resultat que fabriquera le noeud a partir de son test , si il n
        'est pas le rank 0 */
48     *bissResults , /* Tableau final contenant tous les resultats des tests , dont
        l'indice indique le rank du sender du test */
49     **benchResults; /* Idem mais a deux dimensions : en y le sender , en x le
        receveur. Ne sert aussi que pour le rank 0 */
50     StatsResult
51     latencyStats , /* Pointeurs vers les benchs ayant enregistres les latences
        min et max, ainsi que la somme de toutes les latences et la moyenne */
52     flowStats; /* Idem pour les debits */
53     char
54     yamlFile[50]; /* Nom du fichier qui accueillera la sortie YAML si l'option
        -o est passee */
55
56
57     /* Initialisation des connexions MPI et recuperation du nb de noeuds concernees
        par l'execution
58     ainsi que le numero - rang - du noeud courant courant */
59     MPI_Init(&argc , &argv);
60     MPI_Comm_size(MPI_COMM_WORLD, &nbNodes);
61     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
62
63     /* Prise en compte des differentes options passees au script */
64     initOptions(argc , argv , nbNodes , rank , &pktSize , &nbRetry , &bissection , &
        randBiss , &gnuplot , &yaml , yamlFile);
65
66     /* Le buffer sert pour envoyer ou recevoir le mot qui sert de test au debit */
67     buffer = (int *) malloc(sizeof(int)*pktSize);
68     l = 0;
69
70     if(buffer == NULL) {
71         fprintf(stderr , "ERROR: Can't allocate memory.");
72         exit(1);
73     }
74
75     /* Creation des structures qui pourront dorenavant transiter d'un noeud a l'
        autre avec MPI */
76     createBenchType();
77     createTestType();
78     createResultType();
79
80     /* Le MASTER est le rank 0 , c'est lui qui enverra les tests , qui recevra les
        resultats et qui les affichera.
81     Il ne fait pas parti des tests. */
82     if(rank == MASTER) {
83
84         /* Le mode bissection (option -b) consiste a creer des paires de noeuds
            dans ceux faisant parti de l'execution du programme
85         et faire envoyer un mot d'un noeud a l'autre , en demarrant tous en meme tps
            */
86         if(bissection) {
87             bissTests = (YourTest*) malloc(sizeof(YourTest)*nbNodes);
88             bissResults = (MyResult*) malloc(sizeof(MyResult)*nbNodes);
89
90             if(bissTests == NULL || bissResults == NULL) {
91                 fprintf(stderr , "ERROR: Can't allocate memory.");
92                 exit(1);
93             }
94
95             /* Si l'option -rb est active , alors les formations de paires se feront
                aleatoirement parmi les noeuds dispo */
96             if(randBiss)
97                 bissPrepareAllRandTests(bissTests , nbNodes);
98
99             /* Sinon , elles se font en coupant le nombre de noeuds en deux et en
                prenant le premier de la premiere moitie et le
100            premier de la seconde , et ainsi de suite */
101             else
102                 bissPrepareAllTests(bissTests , nbNodes);

```

```

103
104     /* Transmission des tests prepares aux interessees : c'est une fonction
105        collective, ce qui signifie que tous les noeuds
106        doivent avoir execute cette meme fonction, et etre en attente de cet
107        appel du rank 0 (sinon, ils seront attendus).
108        Une fois que chacun a reçu son test, chacun connait son role dans la
109        transaction : envoyeur ou receveur. Chacun se met
110        en ecoute du rank 0 pour savoir quand il devra commencer a jouer son
111        role. */
112     bissTransmitAllTests(bissTests, &myTest);
113
114     /* Chaque noeud est en ecoute de cette meme fonction (puisque c'est
115        aussi une fonction collective). Elle envoie un mot vide
116        en broadcast a tous pour leur dire que le jeu debute : les receveurs se
117        mettent a ecouter les envoyeurs, et les envoyeurs
118        envoient aux receveurs. Chacun se remet ensuite en ecoute du rank 0, et
119        ce jusqu'a ce que la precision imposee par nbRetry
120        soit atteinte. */
121     for(i = 0; i < nbRetry; i++)
122         bissLaunchAllTests();
123
124     /* Derniere fonction collective : tout le monde envoie son resultat,
125        seuls les resultats des envoyeurs auront de l'interet */
126     bissTransmitAllResults(bissResults, &myResult);
127
128     /* Le contraire du mode bisection est le mode matrice : chaque noeud
129        enverra un noeud a tous les autres, afin de pouvoir creer
130        une matrice complete des performances entre tous les noeuds concerne par l'
131        execution, dans tous les sens */
132 } else {
133
134     benchResults = (MyResult**) malloc(sizeof(MyResult)*nbNodes);
135
136     if(benchResults == NULL) {
137         fprintf(stderr, "ERROR: Can't allocate memory.");
138         exit(1);
139     }
140
141     for(i = 0; i < nbNodes; i++) {
142         benchResults[i] = (MyResult*) malloc(sizeof(MyResult)*nbNodes);
143
144         if(benchResults[i] == NULL) {
145             fprintf(stderr, "ERROR: Can't allocate memory.");
146             exit(1);
147         }
148     }
149
150     for(sender = 1; sender < nbNodes; sender++) {
151
152         /* Preparation des tests pour le noeud envoyeur : mise en ecoute de
153            ce noeud, pour tous les autres
154            noeuds qui recevront tour a tour un mot de lui. Chacun de ces
155            noeuds recoit donc un test de receveur */
156         prepareTests(nbNodes, sender);
157
158         for(recver = 1; recver < nbNodes; recver++) {
159             if(recver != sender) {
160
161                 /* Envoi d'un test a l'envoyeur, lui indiquant de
162                    communiquer avec le receveur courant */
163                 launchTests(sender, recver);
164
165                 /* Reception du resultat du test, directement dans la
166                    matrice des resultats */
167                 receiveResults(&benchResults[sender][recver], sender);
168             }
169         }
170     }
171
172     /* Calcul des statistiques */
173     stats(benchResults, bissResults, &latencyStats, &flowStats, nbNodes,
174         bisection);

```

```

161
162     /* Ecriture d'un fichier YAML si l'option -o est passee */
163     if(yaml)
164         toYAML(benchResults, bissResults, yamlFile, nbNodes, bissection);
165
166     /* Sortie en coordonnees pour un graphique gnuplot des debits selon des
167        bisections */
168     if(gnuplot) {
169         toGnuplot(&flowStats, nbNodes);
170
171     /* Sinon matrice + stats */
172     } else {
173
174         /* Affichage d'un tableau/matrice non-parsable sur la sortie standard
175            */
176         displayTab(benchResults, bissResults, bissection, nbNodes);
177
178         /* Affichage des statistiques sur la sortie standard */
179         displayStats(benchResults, bissResults, &latencyStats, &flowStats,
180                    nbNodes, bissection);
181     }
182
183     /* Si le noeud qui execute le programme n'est pas le MASTER (rank != 0), alors
184        il sera charge de participer aux tests
185        qui lui enverra le MASTER, et de lui en renvoyer les resultats.
186        Dans le cas d'une bisection, chaque noeud n'aura qu'un seul role dans sa vie (
187        envoyeur ou receveur), alors que dans le
188        cas de matrice, chacun des noeuds autant de fois envoyeur qu'il y a de noeud,
189        et autant de fois receveur. A l'exclusion,
190        chaque fois, du rank 0, d'ou le -2 */
191     } else while(1++ < (bissection ? 1 : nbNodes*2-4)) {
192
193         sameBenchs = (Bench *) malloc(sizeof(Bench)*nbRetry);
194
195         if(sameBenchs == NULL) {
196             fprintf(stderr, "ERROR: Can't allocate memory.");
197             exit(1);
198         }
199
200         /* Si c'est une bisection, la fonction collective est utilise pour
201            recevoir le test en meme tps que tout le monde */
202         if(bissection)
203             bissTransmitAllTests(bissTests, &myTest);
204
205         /* Sinon, le noeud est simplement en ecoute d'un test sur le rank 0 */
206         else
207             waitTests(&myTest);
208
209         /* Formatage du type MyResult qui sera renvoye, en renseignant le hostname
210            du noeud courant, les deux joueurs de ce tests
211            et les valeurs du flow et debit initialisees a -1 */
212         formatTestsResult(&myResult, &myTest, rank);
213
214         switch(myTest.role) {
215
216             /* Le test recu du rank 0 designe le noeud temporairement comme
217                envoyeur */
218             case SENDER :
219
220                 /* Les tests avec le noeud receveur se repeteront autant de fois
221                    que l'indication de precision
222                    nbRetry l'impose */
223                 for(i = 0; i < nbRetry; i++) {
224
225                     /* Si c'est une bisection, la fonction collective de lancement
226                        des tests est rappelée a chaque fois.
227                        Ceci permet d'etre assure que tout le monde recommence bien son
228                        test au meme moment. Sans cela, les couples
229                        qui auraient pris du retard sur le premier test se retrouveront
230                        seuls dans les derniers tests, lorsque les
231                        plus rapides les auront tous finis. Il seront donc moins
232                        ralentis pour ces derniers tests, qui fausseront
233                        leur moyenne. */

```

```

220         if(bissection)
221             bissLaunchAllTests();
222
223         /* Envoi du mot vide pour la latence , reception du resultat ,
224            envoi du mot de pktSize octets pour le debit ,
225            reception du resultat .
226            Les differences de temps entre chaque envoi et reponse
227            permettent de calculer la latence et le debit , qui
228            seront stockes dans le tableau des benchs de ce test , passe en
229            parametre en ecriture. */
230         benchTests(&myTest , &sameBenchs[i] , pktSize);
231     }
232
233     sumLatency = sumFlow = 0;
234
235     /* Calcul des sommes pour etablir une moyenne de tous les resultats
236        du meme test */
237     for(i = 0; i < nbRetry; i++) {
238         sumLatency += sameBenchs[i].latency;
239         sumFlow += sameBenchs[i].flow;
240     }
241
242     /* Calcul des moyennes et initialisation des valeurs du MyResult
243        qui sera renvoye au MASTER */
244     myResult.result.latency = sumLatency / nbRetry;
245     myResult.result.flow = sumFlow / nbRetry;
246
247     /* Si ca n'est pas une bissection , renvoi direct des resultats au
248        MASTER */
249     if(!bissection)
250         sendResults(&myResult);
251
252     break;
253
254     /* Le role du desactive intervient dans un unique cas de figure : en
255        mode bissection , si le nombre de noeud , en l'enlevant le rank 0 ,
256        est impair. Le noeud du rank le plus eleve recoit donc ce role , qui le
257        fera participer aux fonction collectives , mais qui ne sera pas
258        pris en compte. Tous les noeuds ne naissent pas libres et egaux... */
259     case DEACTIVATED :
260
261         /* Cas d'un receveur */
262         case RECVER :
263
264             /* Le receveur recevra autant de fois que nbRetry l'impose , parce
265                que l'envoyeur enverra tout autant de fois */
266             for(i = 0; i < nbRetry; i++) {
267
268                 /* Si c'est une bissection , la reception est bloquee tant qu'un
269                    nouveau depart de synchro n'a pas ete donne
270                    par le MASTER */
271                 if(bissection)
272                     bissLaunchAllTests();
273
274                 /* Si on est pas dans le cas d'un exclu , reponse aux deux tests
275                    successifs de l'envoyeur partenaire */
276                 if(myTest.role != DEACTIVATED)
277                     responsesToTests(&myTest , pktSize);
278             }
279
280         /* Dans le cas d'une bissection , tous les resultats sont envoyes en meme
281            temps au MASTER, a travers une fonction collective.
282            Les resultats des receveurs ou du desactive ne seront pas pris en compte.
283            */
284         if(bissection)
285             bissTransmitAllResults(bissResults , &myResult);
286     }
287
288     //free(buffer , bissResults , bissTests , benchResults , sameBenchs , nbNodes ,
289           nbRetry);
290
291     MPI_Finalize();

```

```

279
280     return 0;
281 }
282
283
284 /*****
285  *****/
286 /*****
287
288 /*
289  * Gestion des options du script
290  */
291 void initOptions(int argc, char** argv, int nbNodes, int rank, int* pktSize, int*
nbRetry, int* bisection, int* randBiss, int* gnuplot, int* yaml, char*
yamlFile) {
292     char opt, unit;
293
294     /* La taille par default du mot envoye pour les tests de debit est 1M */
295     *pktSize = 1024 * 1024;
296
297     /* Le nombre de fois par default qu'un test est repete pour ameliorer la
precision des resultats est 10 */
298     *nbRetry = 10;
299
300     /* Par default, la bisection (et a fortiori la bisection aleatoire) ainsi que
le yml sont desactives */
301     *bisection = *randBiss = *gnuplot = *yaml = 0;
302
303     while((opt = getopt(argc, argv, "hs:p:bro:g")) != -1) {
304         switch(opt) {
305
306             /* Help */
307             case 'h' :
308                 if(rank == MASTER) {
309                     puts("\nOAR FLOW LATENCY TESTS");
310                     puts("-----");
311                     puts("\t-s <n>K      : Message size for flow tests exchanges, in
bytes (with K, M, or G suffix). Min 64K, default 1M.");
312                     puts("\t-p <n>      : Tests precision (repeats each test <n>
times and make averages). Default 10.");
313                     puts("\t-b          : Bisection (first node of the first half
with the first node of the seconde half, and so on).");
314                     puts("\t-rb, -r    : Bisection, with random pairs.");
315                     puts("\t-bg, -g    : Bisection, with gnuplot coordinates output.
");
316                     puts("\t-o <file> : YAML output.\n");
317                     puts("\t-h          : This help.\n");
318                     puts("AUTHORS : <julien@vaubourg.com>\n          <sebastien.
badia@gmail.com>\n");
319                 }
320
321                 MPI_Finalize();
322                 exit(0);
323                 break;
324
325             /* Taille des mots qui seront envoyes pour les tests de debit, en
octets et avec un suffix (K, M, G) */
326             case 's' :
327
328                 /* Recuperation de l' unite utilisee en recuperant le dernier
caractere de la valeur (nK, nM, ou nG), puis suppression de
celle-ci */
329                 unit = optarg[strlen(optarg) - 1];
330                 *(strchr(optarg, unit)) = '\0';
331                 *pktSize = atoi(optarg);
332
333                 /* En l'absence de break intermediaires, la taille du mot sera
multipliee par autant de fois qu'il faudra traverser une unite
subalterne pour atteindre le break final. */
334                 switch(unit) {
335                     case 'G' :
336                         *pktSize *= 1024;
337                     case 'M' :

```

```

339         *pktSize *= 1024;
340     case 'K' :
341         *pktSize *= 1024;
342     break;
343
344     default :
345         if(rank == MASTER)
346             fprintf(stderr, "ERROR: Unit -s unknown (K, M, or G).")
347                 ;
348
349         exit(1);
350     }
351
352     /* Un calcul de debit ne peut pas se faire avec un mot de moins de
353        64K (tests NWS) */
354     if(*pktSize < 64*1024) {
355         if(rank == MASTER)
356             fprintf(stderr, "ERROR: For realistic flows results, the
357                 message size defined by -s must be greater than 64KB (
358                 NWS method).");
359
360         exit(1);
361     }
362     break;
363
364     /* Mode bisection : couples aleatoires si -r, en fonction de la
365        moitie de la liste des noeuds sinon */
366     case 'r' :
367         *randBiss = 1;
368     case 'b' :
369         *bisection = 1;
370     break;
371
372     /* Sortie pour un graphique gnuplot */
373     case 'g' :
374         *gnuplot = 1;
375         *bisection = 1;
376     break;
377
378     /* Sortie dans un fichier YAML en plus de la sortie matrice */
379     case 'o' :
380         *yaml = 1;
381         strncpy(yamlFile, optarg, 50);
382     break;
383
384     /* Les tests se feront autant de fois que l'indicateur de precision -p
385        le dit, en prenant la moyenne des resultats de tous */
386     case 'p' :
387         *nbRetry = atoi(optarg);
388
389         if(*nbRetry < 0) {
390             if(rank == MASTER)
391                 fprintf(stderr, "ERROR: The -p option must be positive.");
392
393             exit(1);
394         }
395     break;
396
397     /* Option inconnue ou mal renseignee */
398     case '?' :
399         if(rank == MASTER) {
400             if(optopt == 's' || optopt == 'r' || optopt == 'o')
401                 fprintf(stderr, "ERROR: The -%c option require an argument
402                     .\n", optopt);
403             else if(isprint(optopt))
404                 fprintf(stderr, "ERROR: The -%c option is unknown.\n",
405                     optopt);
406             else
407                 fprintf(stderr, "ERROR: The '\\x%x' character is unknown.\n
408                     ", optopt);
409         }
410     default :
411         exit(1);

```

```

403     }
404 }
405
406 if (nbNodes < 3) {
407     if (rank == MASTER)
408         fprintf(stderr, "ERROR : Tests requires at least 3 nodes (1 for
409             monitoring, at least 2 for exchanges).");
410
411     exit(1);
412 }
413
414 /*
415  * Creation d'un type BenchType pour MPI qui permettra de faire transiter des
416  * structures Bench d'un noeud a l'autre.
417 */
418 void createBenchType() {
419     MPI_Type_extent(MPI_INT, &extentType);
420
421     benchTypeDisp[0] = 0;
422     benchTypeDisp[1] = extentType;
423     benchTypeDisp[2] = extentType*2;
424
425     MPI_Type_extent(MPI_FLOAT, &extentType);
426
427     benchTypeDisp[3] = benchTypeDisp[2] + extentType;
428
429     MPI_Type_struct(4, benchTypeBlocks, benchTypeDisp, benchTypeTypes, &BenchType);
430     MPI_Type_commit(&BenchType);
431 }
432
433 /*
434  * Creation d'un type TestType pour MPI qui permettra de faire transiter des
435  * structures YourTest d'un noeud a l'autre.
436 */
437 void createTestType() {
438     MPI_Type_extent(MPI_INT, &extentType);
439
440     testTypeDisp[0] = 0;
441     testTypeDisp[1] = extentType;
442
443     MPI_Type_struct(2, testTypeBlocks, testTypeDisp, testTypeTypes, &TestType);
444     MPI_Type_commit(&TestType);
445 }
446
447 /*
448  * Creation d'un type ResultType pour MPI qui permettra de faire transiter des
449  * structures MyResult d'un noeud a l'autre.
450 */
451 void createResultType() {
452     MPI_Datatype resultTypeTypes[2] = { MPI_CHAR, BenchType };
453     MPI_Type_extent(MPI_CHAR, &extentType);
454
455     resultTypeDisp[0] = 0;
456     resultTypeDisp[1] = extentType*100;
457
458     MPI_Type_struct(2, resultTypeBlocks, resultTypeDisp, resultTypeTypes, &
459         ResultType);
460     MPI_Type_commit(&ResultType);
461 }
462
463 /*
464  * Met en ecoute tous les autres noeuds que l'envoyeur designe, afin qu'ils soient
465  * prêts a recevoir un test de celui-ci.
466 */
467 void prepareTests(int nbNodes, int sender) {
468     int i;
469     YourTest t;
470
471     for(i = 1; i < nbNodes; i++) {
472         if(i != sender) {
473             t.role = RECV;
474             t.withRank = sender;
475         }
476     }
477 }

```

```

470
471     MPI_Send(&t, 1, TestType, i, 0, MPI_COMM_WORLD);
472 }
473 }
474 }
475
476 /*
477  * Envoi de son test a l'envoyeur pour lui indiquer de communiquer avec un des
478  * receveurs pret a receptionner un mot
479  */
480 void launchTests(int sender, int recver) {
481     YourTest t;
482
483     t.role = SENDER;
484     t.withRank = recver;
485
486     MPI_Send(&t, 1, TestType, sender, 0, MPI_COMM_WORLD);
487 }
488
489 /*
490  * Par default, tous les noeuds attendent un test du MASTER.
491  */
492 void waitTests(YourTest* t) {
493     MPI_Recv(t, 1, TestType, MASTER, 0, MPI_COMM_WORLD, &status);
494 }
495
496 /*
497  * Envoi du resultat MyResult au MASTER, de la part de l'envoyeur.
498  */
499 void sendResults(MyResult* r) {
500     MPI_Send(r, 1, ResultType, MASTER, 0, MPI_COMM_WORLD);
501 }
502
503 /*
504  * Reception du resultat MyResult de l'envoyeur qui vient de realiser son test (
505  * pour le rank 0).
506  */
507 void receiveResults(MyResult* r, int sender) {
508     MPI_Recv(r, 1, ResultType, sender, 0, MPI_COMM_WORLD, &status);
509 }
510 /* Formatage du MyResult qui sera renvoye au MASTER */
511 void formatTestsResult(MyResult* r, YourTest* t, int rank) {
512     int sizeHostname;
513     char* sep;
514
515     /* Les deux joueurs sont renseignes, et la latence est mise a -1 : si le test n
516     'est
517     jamais utilise, la latence restera ainsi, et pourra alors etre un indicateur de
518     la non
519     pertinence du resultat. */
520     r->result.sender = rank;
521     r->result.recver = t->withRank;
522     r->result.latency = r->result.flow = -1;
523
524     /* Renseignement du nom du noeud courant */
525     MPI_Get_processor_name(r->myHostname, &sizeHostname);
526
527     /* Reduction du nom du noeud a sa premiere partie (on vire .site.grid5000.fr)
528     */
529     sep = strchr(r->myHostname, '.');
530     *sep = '\0';
531 }
532
533 /*
534  * Tests de debit/latence de l'envoyeur vers le receveur.
535  */
536 void benchTests(YourTest* t, Bench* r, int pktSize) {
537     double start, stop;
538
539     /* Un mot vide (4 octets) est envoye au receveur, qui repondra immediatement un
540     mot de la meme nature.

```

```

537     Le temps est compte, de l'envoi du mot au receveur jusqu'a la reception de sa
        reponse. */
538     start = MPI_Wtime();
539     MPI_Send(buffer, 0, MPI_BYTE, t->withRank, 1, MPI_COMM_WORLD);
540     MPI_Recv(buffer, 0, MPI_BYTE, t->withRank, 1, MPI_COMM_WORLD, &status);
541     stop = MPI_Wtime();
542
543     /* La latence est calculee en fonction du temps mis par le mot pour arriver au
        destinataire (division par 2
544     pour eliminer le tps de reponse qui est cense etre identique.
545     Passage de seconde a microseconde en multipliant par un million. */
546     r->latency = ((stop-start) / 2) * 1e6;
547
548     /* Un mot plus ou moins consequent (option -s) est envoye au receveur. Celui-ci
        renvoyant un mot vide.
549     Le temps est egalement compte, de l'envoi jusqu'a la reception. */
550     start = MPI_Wtime();
551     MPI_Send(buffer, pktSize, MPI_BYTE, t->withRank, 1, MPI_COMM_WORLD);
552     MPI_Recv(buffer, 0, MPI_BYTE, t->withRank, 1, MPI_COMM_WORLD, &status);
553     stop = MPI_Wtime();
554
555     /* La difference de temps est prise en compte. On lui soustrait deux fois la
        latence - en seconde grace a la
556     division - d'un mot vide (soit le tps que met le tuyau a faire transiter les
        donnees, quelque soit la taille des
557     donnees). Enfin, on divise le nombre d'octets qui ont transites avec ce
        resultat. On divise le tout par 1024 au
558     carre pour avoir des Mo/s au lieu de o/s. Cette facon de proceder releve de l'
        approche NWS : http://nws.cs.ucsb.edu */
559     r->flow = pktSize / ((stop-start) - (2*r->latency/1e6)) / pow(1024, 2);
560 }
561
562 /*
563 * Reponses automatiques aux envois du noeud qui joue le role d'envoyeur.
564 */
565 void responsesToTests(YourTest* t, int pktSize) {
566     MPI_Recv(buffer, 0, MPI_BYTE, t->withRank, 1, MPI_COMM_WORLD, &status);
567     MPI_Send(buffer, 0, MPI_BYTE, t->withRank, 1, MPI_COMM_WORLD);
568
569     MPI_Recv(buffer, pktSize, MPI_BYTE, t->withRank, 1, MPI_COMM_WORLD, &status);
570     MPI_Send(buffer, 0, MPI_BYTE, t->withRank, 1, MPI_COMM_WORLD);
571 }
572
573 /*
574 * Creation de tous les tests qui seront repartis entre tous les noeuds, pour le
        mode bisection, sans les envoyer et en les
575 * stockant dans un tableau.
576 */
577 void bissPrepareAllTests(YourTest* bissTests, int nbNodes) {
578     int m, i;
579     YourTest t;
580
581     /* Le rank 0 ne participe pas */
582     nbNodes--;
583
584     /* Si il y a un nb impaire de noeud, on ignore le dernier, en le faisant quand
        meme participer
585     aux fonctions collectives*/
586     if(nbNodes % 2) {
587         bissTests[nbNodes].role = DEACTIVATED;
588         bissTests[nbNodes].withRank = -1;
589         nbNodes--;
590     }
591
592     m = nbNodes / 2;
593
594     /* Pour chaque paire */
595     for(i = 1; i <= m; i++) {
596
597         /* Le receveur de la paire est le n ieme de la seconde moitie */
598         bissTests[m+i].role = RECEIVER;
599         bissTests[m+i].withRank = i;
600

```

```

601     /* L'envoyeur est le n ieme de la premiere moitie */
602     bissTests[i].role = SENDER;
603     bissTests[i].withRank = m+i;
604 }
605 }
606
607 /*
608  * Creations des tests pour la bisection , en creant les paires de noeuds de facon
609  * aleatoire (sans envoyer les tests ,
610  * et en les stockant dans un tableau .
611  */
612 void bissPrepareAllRandTests(YourTest* bissTests , int nbNodes) {
613     int i , r , randRank , ranks[2] , iLeft , *ranksLeft , nbLeft;
614     YourTest t;
615
616     /* Le rank 0 ne participe pas */
617     nbNodes--;
618
619     /* Si il y a un nb impaire de noeud , on ignore le dernier , en le faisant quand
620     meme participer
621     aux fonctions collectives*/
622     if(nbNodes % 2) {
623         bissTests[nbNodes].role = DEACTIVATED;
624         bissTests[nbNodes].withRank = -1;
625         nbNodes--;
626     }
627
628     /* Nombre et liste des noeuds qui n'ont pas encore de test attribue */
629     nbLeft = nbNodes;
630     ranksLeft = (int*) malloc(sizeof(int) * nbLeft);
631
632     if(ranksLeft == NULL) {
633         fprintf(stderr , "ERROR: Can't allocate memory.");
634         exit(1);
635     }
636
637     /* Initialisation des noeuds : avec la valeur 1 , chacun est encore vaquant */
638     for(i = 0; i < nbLeft; ranksLeft[i++] = 1);
639
640     /* Pour tous les noeuds vaquants restants */
641     while(nbLeft != 0) {
642
643         /* Extraction de deux noeuds choisis aleatoirement parmi les noeuds encore
644         disponibles
645         Premier tour pour l'envoyeur , second tour pour le receveur */
646         for(i = 0; i < 2; i++) {
647             srand(time(NULL));
648
649             /* Tirage d'un nombre aleatoire qui definira le rank du noeud choisi ,
650             parmi les ranks restants .
651             Ex: si 3 est tire , le rank selectionne sera le 3ieme qui n'a pas encore
652             ete utilise */
653             randRank = rand() % (nbLeft - 1);
654
655             r = -1;
656             iLeft = 0;
657             ranks[i] = -1;
658
659             /* Pour tous les ranks jusqu'a ce que le n ieme rank encore vaquant
660             soit atteint */
661             while(ranks[i] == -1) {
662
663                 /* Si le rank en cours est encore vaquant */
664                 if(ranksLeft[i+r] != -1) {
665
666                     /* Si le rank en cours correspond au rank du tirage aleatoire ,
667                     selection de ce rank (+1 , puisque le rank 0 a ete supprime)
668                     */
669                     if(iLeft == randRank)
670                         ranks[i] = r + 1;
671
672                     /* Sinon incrementation du nb des ranks encore vaquants deja
673                     rencontres */

```

```

665         iLeft++;
666     }
667 }
668
669     /* Ce rank ne pourra plus etre selectionne */
670     ranksLeft[r] = -1;
671 }
672
673     /* Deux ranks en moins a attribuer */
674     nbLeft -= 2;
675
676     /* Creation du test du receveur, qui attendra alors un message de son
677     camarade de jeu */
678     bissTests[ranks[1]].role = RECV;
679     bissTests[ranks[1]].withRank = ranks[0];
680
681     /* Creation du test de l'envoyeur */
682     bissTests[ranks[0]].role = SENDER;
683     bissTests[ranks[0]].withRank = ranks[1];
684 }
685 }
686
687 /*
688 * Fonction collective d'envoi du test associe a chaque rank, depuis le MASTER.
689 */
690 void bissTransmitAllTests(YourTest* bissTests, YourTest* t) {
691     MPI_Scatter(bissTests, 1, TestType, t, 1, TestType, MASTER, MPI_COMM_WORLD);
692 }
693
694 /*
695 * Fonction collective d'envoi d'un broadcast vide, permettant de lancer un depart
696 synchro des tests de bisections.
697 */
698 void bissLaunchAllTests() {
699     MPI_Bcast(buffer, 0, MPI_BYTE, MASTER, MPI_COMM_WORLD);
700 }
701
702 /*
703 * Fonction collective d'envoi des resultats de la part de chaque rank vers le
704 MASTER.
705 */
706 void bissTransmitAllResults(MyResult* bissResults, MyResult* r) {
707     MPI_Gather(r, 1, ResultType, bissResults, 1, ResultType, MASTER, MPI_COMM_WORLD);
708 }
709
710 /* Fonction permettant la conversion numero de rank vers Hostname. */
711 char* rankToHostname(MyResult** r, MyResult* rBiss, int rank, int bissection) {
712     if(bissection)
713         return rBiss[rank].myHostname;
714     else
715         return r[rank][rank == 1 ? 2 : 1].myHostname;
716 }
717
718 /*
719 * Affichage des resultats sous forme de tableau texte non-parsable.
720 */
721 void displayTab(MyResult** r, MyResult* rBiss, int bissection, int nbNodes) {
722     int x, y, i;
723
724     /* Affichage des resultats de la bisections sous forme d'un tableau simple
725     dont la premier colonne indique
726     l'envoyeur et la seconde le receveur avec la latence et le debit calculees */
727     if(bissection) {
728
729         /* Pour chaque envoyeur */
730         for(i = 1; i < nbNodes; i++) {
731
732             /* La latence differente de -1 signifie que le MyResult a ete exploite
733             et n'est pas le retour d'un
734             RECV ou d'un DEACTIVATED */
735             if(rBiss[i].result.latency != -1) {
736                 puts("+-----+");
737             }
738         }
739     }

```

```

732         printf("| From %-15s", rankToHostname(r, rBiss, i, bisection));
733         printf("| To %-17s |\n", rankToHostname(r, rBiss, rBiss[i].result.
            receiver, bisection));
734         printf(" |                               ");
735         printf(" | %17.3f us |\n", rBiss[i].result.latency);
736         printf(" |                               ");
737         printf(" | %15.3f Mo/s |\n", rBiss[i].result.flow);
738     }
739 }
740
741 puts("+-----+");
742
743 /* Affichage des resultats sous forme d'une matrice equilibree de tous les
744    noeuds, avec la debit et la latence pour chaque
745    paire de noeud possible, dans les deux sens */
746 } else {
747     /* Premiere ligne de tirets */
748     printf("                               ");
749     for(x = 1; x < nbNodes; printf("+-----"), x++);
750     printf("+\n                               ");
751
752     /* Lignes des entetes de colonnes (receveurs) */
753     for(x = 1; x < nbNodes; printf("| To %-17s ", rankToHostname(r, rBiss, x++,
            bisection)));
754     printf("|\n");
755
756     /* Ligne de tirets de separation */
757     for(x = 0; x < nbNodes; printf("+-----"), x++);
758     puts("+");
759
760     /* Pour chaque ligne/envoyeur */
761     for(y = 1; y < nbNodes; y++) {
762
763         /* Entete */
764         printf("| From %-15s ", r[y][y == 1 ? 2 : 1].myHostname);
765
766         /* Latence, ou suite de tirets si on est l'envoyeur correspond au
767            receveur */
768         for(x = 1; x < nbNodes; x++) {
769             (x == y) ?
770                 printf(" |-----" ) :
771                 printf(" | %17.3f us ", r[y][x].result.latency);
772         }
773         printf(" |\n |                               ");
774
775         /* Debit, ou suite de tirets si on est l'envoyeur correspond au
776            receveur */
777         for(x = 1; x < nbNodes; x++) {
778             (x == y) ?
779                 printf(" |-----" ) :
780                 printf(" | %15.3f Mo/s ", r[y][x].result.flow);
781         }
782
783         /* Fermeture de ligne */
784         puts("|");
785         for(x = 1; x < nbNodes+1; printf("+-----"), x++);
786         puts("+");
787     }
788 }
789 /*
790 * Affichage des statistiques en relation avec la matrice ou la bisection sous
791 * forme de texte non-parsable.
792 * Latence min et max ainsi que debit min et max.
793 * Ainsi que les sommes et les moyennes.
794 */
795 void displayStats(MyResult** r, MyResult* rBiss, StatsResult* latencyStats,
796                 StatsResult* flowStats, int nbNodes, int bisection) {
797     puts("\nLatency :");

```

```

798     /* Latence Maximum et affichage de la paire de noeuds correspondante, en
799     hostname */
800     printf(
801         "Max : %.3f us \tFrom %s to %s\n",
802         latencyStats->max->latency,
803         rankToHostname(r, rBiss, latencyStats->max->sender, bissection),
804         rankToHostname(r, rBiss, latencyStats->max->recver, bissection)
805     );
806     /* Latence Minimum et affichage de la paire de noeuds correspondante, en
807     hostname */
808     printf(
809         "Min : %.3f us \tFrom %s to %s\n",
810         latencyStats->min->latency,
811         rankToHostname(r, rBiss, latencyStats->min->sender, bissection),
812         rankToHostname(r, rBiss, latencyStats->min->recver, bissection)
813     );
814     /* Somme des latences et moyenne */
815     printf("Sum : %.3f us\n", latencyStats->sum);
816     printf("Avg : %.3f us\n", latencyStats->avg);
817
818     puts("\nFlow :");
819
820     /* Debit Maximum et affichage de la paire de noeuds correspondante, en hostname
821     */
822     printf(
823         "Max : %.3f Mo/s \tFrom %s to %s\n",
824         flowStats->max->flow,
825         rankToHostname(r, rBiss, flowStats->max->sender, bissection),
826         rankToHostname(r, rBiss, flowStats->max->recver, bissection)
827     );
828     /* Debit Minimum et affichage de la paire de noeuds correspondante, en hostname
829     */
830     printf(
831         "Min : %.3f Mo/s \tFrom %s to %s\n",
832         flowStats->min->flow,
833         rankToHostname(r, rBiss, flowStats->min->sender, bissection),
834         rankToHostname(r, rBiss, flowStats->min->recver, bissection)
835     );
836     /* Somme des debits et moyenne */
837     printf("Sum : %.3f Mo/s\n", flowStats->sum);
838     printf("Avg : %.3f Mo/s\n", flowStats->avg);
839 }
840
841 /*
842 * Ecriture en YAML dans un fichier des resultats.
843 */
844 void toYAML(MyResult** r, MyResult* rBiss, char* yamlFile, int nbNodes, int
845 bissection) {
846     FILE* yaml;
847     int x, y, i;
848
849     yaml = fopen(yamlFile, "w");
850
851     if(yaml == NULL) {
852         fprintf(stderr, "ERROR: Can't write the yaml file.");
853         return;
854     }
855
856     /* Export du tableau des resultats (meme structure que pour la matrice) */
857     if(bissection) {
858         /* Pour chaque envoyeur */
859         for(y = 1; y < nbNodes; y++) {
860
861             /* Resultat d'un envoyeur */
862             if(rBiss[y].result.latency != -1) {
863                 fprintf(yaml, "%s\n", rankToHostname(r, rBiss, y, bissection));
864                 fprintf(yaml, " %s\n", rankToHostname(r, rBiss, rBiss[y].result.
865                     recver, bissection));

```

```

865         fprintf(yaml, "    latency ; %.3f\n", rBiss[y].result.latency);
866         fprintf(yaml, "    flow : %.3f\n", rBiss[y].result.flow);
867     }
868 }
869
870 /* Export de la matrice des resultats (3 niveaux : envoyeur => receveur =>
871    debit/latence) */
872 } else {
873     /* Pour chaque envoyeur */
874     for(y = 1; y < nbNodes; y++) {
875         fprintf(yaml, "%s :\n", rankToHostname(r, rBiss, y, bisection));
876
877         /* Pour chaque receveur */
878         for(x = 1; x < nbNodes; x++) {
879             if(y != x) {
880                 fprintf(yaml, "  %s :\n", rankToHostname(r, rBiss, x,
881                    bisection));
882                 fprintf(yaml, "    latency : %.3f\n", r[y][x].result.latency);
883                 fprintf(yaml, "    flow : %.3f\n", r[y][x].result.flow);
884             }
885         }
886     }
887 }
888 fclose(yaml);
889 }
890
891 /*
892 * Exporte la somme des debits constates sous forme de coordonnees (x: nb noeuds, y
893    : somme).
894 * Sert pour la construction d'un graphique gnuplot du total des debits de
895    bisections testees avec des nombres de noeuds
896    variables, et dans le cadre d'un script plus global.
897 */
898 void toGnuplot(StatsResult* flowStats, int nbNodes) {
899     printf("%d\t%.3f\n", nbNodes, flowStats->sum);
900 }
901
902 /*
903 * Fonction de calcul des statistiques.
904 */
905 void stats(MyResult** r, MyResult* rBiss, StatsResult* latencyStats, StatsResult*
906    flowStats, int nbNodes, int bisection) {
907     int i, j;
908     Bench initLatencyMin, initLatencyMax, initFlowMin, initFlowMax;
909
910     /* Initialisation de la structure benchesStatsBounds */
911     latencyStats->min = &initLatencyMin;
912     latencyStats->min->latency = 99999999.99;
913
914     latencyStats->max = &initLatencyMax;
915     latencyStats->max->latency = -1;
916
917     flowStats->min = &initFlowMin;
918     flowStats->min->flow = 99999999.99;
919
920     flowStats->max = &initFlowMax;
921     flowStats->max->flow = -1;
922
923     /* Initialisation des sommes */
924     latencyStats->sum = flowStats->sum = 0;
925
926     if(bisection) {
927         for (i = 1; i < nbNodes; i++) {
928
929             /* Si la latence est encore en etat d'initialisation on arrete */
930             if(rBiss[i].result.latency != -1) {
931
932                 /* Calcul de la Latence Minimum et attribution de l'adresse
933                    correspondante */
934                 if (rBiss[i].result.latency < (latencyStats->min->latency)) {
935                     latencyStats->min = &(rBiss[i].result);
936                 }
937             }
938         }
939     }

```

```

932     }
933
934     /* Calcul de la Latence Maximum et attribution de l'adresse
correspondante */
935     if (rBiss[i].result.latency > (latencyStats->max->latency)) {
936         latencyStats->max = &(rBiss[i].result);
937     }
938
939     /* Calcul du debit Minimum et attribution de l'adresse
correspondante */
940     if (rBiss[i].result.flow < (flowStats->min->flow)) {
941         flowStats->min = &(rBiss[i].result);
942     }
943
944     /* Calcul du debit Maximum et attribution de l'adresse
correspondante */
945     if (rBiss[i].result.flow > (flowStats->max->flow)) {
946         flowStats->max = &(rBiss[i].result);
947     }
948
949     /* Calcul de la somme des debits */
950     flowStats->sum += rBiss[i].result.flow;
951
952     /* Calcul de la somme des latence */
953     latencyStats->sum += rBiss[i].result.latency;
954 }
955 }
956
957 /* Calculs des Moyennes */
958
959 /* Debit */
960 flowStats->avg = flowStats->sum / ((nbNodes-1)/2);
961
962 /* Latence */
963 latencyStats->avg = latencyStats->sum / ((nbNodes-1)/2);
964
965 } else {
966
967     /* Double boucle afin de parcourir l'ensemble de la matrice */
968     for (i = 1; i < nbNodes; i++) {
969         for (j = 1; j < nbNodes; ++j) {
970
971             /* On evite le cas particulier ou le rank est egal (case vide) */
972             if (i != j) {
973
974                 /* Calcul de la Latence Minimum et attribution de l'adresse
correspondante */
975                 if (r[i][j].result.latency < (latencyStats->min->latency)) {
976                     latencyStats->min = &(r[i][j].result);
977                 }
978
979                 /* Calcul de la Latence Maximum et attribution de l'adresse
correspondante */
980                 if (r[i][j].result.latency > (latencyStats->max->latency)) {
981                     latencyStats->max = &(r[i][j].result);
982                 }
983
984                 /* Calcul du debit Minimum et attribution de l'adresse
correspondante */
985                 if (r[i][j].result.flow < (flowStats->min->flow)) {
986                     flowStats->min = &(r[i][j].result);
987                 }
988
989                 /* Calcul du debit Maximum et attribution de l'adresse
correspondante */
990                 if (r[i][j].result.flow > (flowStats->max->flow)) {
991                     flowStats->max = &(r[i][j].result);
992                 }
993
994                 /* Calcul de la somme des debits */
995                 flowStats->sum += r[i][j].result.flow;
996
997                 /* Calcul de la somme des latence */

```

```

998         latencyStats->sum += r[i][j].result.latency;
999     }
1000 }
1001 }
1002
1003     /* Calcul des Moyennes et enregistrement */
1004
1005     /* Debit */
1006     flowStats->avg = flowStats->sum / (pow((nbNodes-1),2)-(nbNodes-1));
1007
1008     /* Latence */
1009     latencyStats->avg = latencyStats->sum / (pow((nbNodes-1),2)-(nbNodes-1));
1010 }
1011 }
1012
1013 /*
1014  * Liberation de la memoire pour les allocations faites manuellement.
1015  */
1016 void freee(int* buffer, MyResult* bissResults, YourTest* bissTests, MyResult**
1017     benchResults, Bench* sameBenchs, int nbNodes, int nbRetry) {
1018     int i;
1019
1020     free(buffer);
1021     free(bissResults);
1022     free(bissTests);
1023     free(sameBenchs);
1024
1025     for(i = 0; i < nbNodes; free(benchResults[i++]));
1026     free(benchResults);
1027 }

```

6 Librairie Latence et Débit

```

1  #ifndef LATENCY_FLOW_TESTS
2  #define LATENCY_FLOW_TESTS
3
4  #define DEACTIVATED -1
5  #define RECVER 0
6  #define SENDER 1
7  #define MASTER 0
8
9  typedef struct {
10     int sender, recver;
11     float latency, flow;
12 } Bench;
13
14 typedef struct {
15     int role, withRank;
16 } YourTest;
17
18 typedef struct {
19     char myHostname[100];
20     Bench result;
21 } MyResult;
22
23 typedef struct {
24     Bench *min, *max;
25     float sum, avg;
26 } StatsResult;
27
28 MPI_Datatype BenchType, TestType, ResultType;
29 MPI_Datatype benchTypeTypes[4] = { MPI_INT, MPI_INT, MPI_FLOAT, MPI_FLOAT };
30 MPI_Datatype testTypeTypes[2] = { MPI_INT, MPI_INT };
31 MPI_Aint benchTypeDisp[4], testTypeDisp[2], resultTypeDisp[2], extentType;
32 MPI_Status status;
33
34 int benchTypeBlocks[4] = { 1, 1, 1, 1 };
35 int testTypeBlocks[2] = { 1, 1 };
36 int resultTypeBlocks[2] = { 100, 1 };
37 int* buffer;
38
39 void initOptions(int argc, char** argv, int nbNodes, int rank, int* pktSize, int*

```

```

        nbRetry, int* bisection, int* randBiss, int* gnuplot, int* yaml, char*
        yamlFile);
40
41 void createBenchType();
42 void createTestType();
43 void createResultType();
44
45 void prepareTests(int nbNodes, int sender);
46 void launchTests(int sender, int recver);
47 void waitTests(YourTest* t);
48 void sendResults(MyResult* r);
49 void receiveResults(MyResult* r, int sender);
50
51 void formatTestsResult(MyResult* r, YourTest* t, int rank);
52 void benchTests(YourTest* t, Bench* r, int pktSize);
53 void responsesToTests(YourTest* t, int pktSize);
54
55 void bissPrepareAllTests(YourTest* bissTests, int nbNodes);
56 void bissPrepareAllRandTests(YourTest* bissTests, int nbNodes);
57 void bissTransmitAllTests(YourTest* bissTests, YourTest* t);
58 void bissLaunchAllTests();
59 void bissTransmitAllResults(MyResult* bissResults, MyResult* r);
60
61 char* rankToHostname(MyResult** r, MyResult* rBiss, int rank, int bisection);
62
63 void displayTab(MyResult** r, MyResult* rBiss, int bisection, int nbNodes);
64 void toYAML(MyResult** r, MyResult* rBiss, char* yamlFile, int nbNodes, int
        bisection);
65 void toGnuplot(StatsResult* flowStats, int nbNodes);
66
67 void stats(MyResult** r, MyResult* rBiss, StatsResult* latencyStats, StatsResult*
        flowStats, int nbNodes, int bisection);
68 void displayStats(MyResult** r, MyResult* rBiss, StatsResult* latencyStats,
        StatsResult* flowStats, int nbNodes, int bisection);
69
70 void freee(int* buffer, MyResult* bissResults, YourTest* bissTests, MyResult**
        benchResults, Bench* sameBenchs, int nbNodes, int nbRetry);
71
72 #endif

```

7 Script de calcul du débit total en fonction du nombre de nœuds

```

1  #!/ bin / bash
2
3  if [[ -z "$OAR_NODEFILE" ]]; then
4      echo "ERROR : You must have a nodes reservation (oarsub, on a single cluster)."
5      >&2
6      exit 1
7  fi
8
9  if [[ -z $1 ]]; then
10     echo "Usage: ./flowsSumsBis.sh <step>" >&2
11     exit 1
12 fi
13 STEP=$1
14 MACHINES=/tmp/${USER}_machines
15 PLOTFILE=flowsSumsBis.dat
16 i=3
17
18 head -n1 $OAR_NODEFILE > $MACHINES
19 sort -u $OAR_NODEFILE >> $MACHINES
20 NB=$(cat $MACHINES | wc -l)
21 :> $PLOTFILE
22
23 if [[ $NB -lt 3 ]]; then
24     echo "ERROR : Tests requires at least 3 nodes (1 for monitoring, at least 2 for
25         exchanges)." >&2
26     exit 1

```

```

26 fi
27
28 while [[ $i -le $NB ]]; do
29     echo "Bisection : calculating the flow sum with $i nodes..."
30     mpirun -np $i --mca plm_rsh_agent oarsh --machinefile $MACHINES ./
        latency_flow_tests -g 2> /dev/null >> $PLOTFILE
31     i=$(( i + STEP ))
32 done
33
34 rm $MACHINES
35 exit 0

```

8 Script gnuplot pour l'exploitation des résultats du script *bash*

```

1 set terminal postscript eps color "Sans-serif" 16
2 set output 'latencyFlows.eps'
3 set key right bottom
4 set xlabel "Nombre de noeuds dans la bisection"
5 set ylabel "Debit en Mo/s"
6 set grid
7
8 plot 'Result/gdx/latencyFlows.dat' using 1:2 title 'gdx' with linespoints
9 quit

```

9 Script Ruby d'export Yaml vers Html

```

1 #!/usr/bin/ruby -w
2 #
3 # YAML TO HTML (LATENCY FLOW TESTS)
4 #   With great colors !
5 #
6 # OPTIONS : Voir -h
7 #
8 # AUTEURS : <julien@vaubourg.com>
9 #           <seb@sebian.fr> <badia.seb@gmail.com>
10 #
11 # 2010, pour Grid5000
12 #
13
14 # Ce script a pour but de transformer le fichier d'export yaml du programme '
15 #   latency_flow_test'
16 #   en un tableau html, avec des couleurs. (faisant ressortir les min/max ainsi que
17 #   les quartiles)
18
19 # Utilise la classe yaml, il sera peut etre necessaire de l'installer
20 require 'yaml'
21 require 'getoptlong'
22
23 # Les options -i pour input et -o pour output sont indispensables.
24 opts = GetoptLong.new(
25   [ '--help', '-h', GetoptLong::NO_ARGUMENT ],
26   [ '--yaml', '-i', GetoptLong::REQUIRED_ARGUMENT ],
27   [ '--html', '-o', GetoptLong::REQUIRED_ARGUMENT ]
28 )
29
30 fileyaml = html = ''
31
32 opts.each do |opt, arg|
33   case opt
34     when '--help'
35       puts "\nYAML TO HTML"
36       puts "-----"
37       puts "This script allows the conversion of YAML file"
38       puts "(output file of latency_flow_test) into an HTML table."
39       puts "An HTML table was easily viewable on a website.\n"
40       puts "The color code allows to visualize at once the most low knots,"
41       puts "and those them more successful"

```



```

110 q1Latency = latencys.last[:ecc] / 4
111 q3Latency = q1Latency * 3
112 q1Flow = flows.last[:ecc] / 4
113 q3Flow = q1Flow * 3
114
115 # On cree un variable 'matrix' dans laquelle on concatenera tout le html genere
116 matrix = "<tr><th></th><th class=\"bot\">#{ benchResults.keys.join('</th><th class="
117   "bot">') }</th><th class=\"bot\"></th></tr>"
118
119 benchResults.each do |sender, recvers|
120   matrix += '<tr><th class="right' + (line == benchResults.size - 1 ? " bot" : ""
121     ) + '" rowspan="2">>' + sender + '</th>\'
122   # On remplit la matrix avec tout les debits
123   column = 0
124   recvers.each do |recver, results|
125     matrix += '<td></td>' if line == column
126     column += 1
127
128     matrix += <<-HTML
129     <td class="flow#{
130       case results['flow']
131         when flows.first[:flow]
132           " min"
133         when flows.last[:flow]
134           " max"
135         else
136           case flows[flows.index { |x| x[:sender] == sender && x[:recver]
137             == recver }][:ecc]
138             when 0..q1Flow
139               " low"
140             when q3Flow..flows.last[:ecc]
141               " high"
142           end
143         end
144       }">#{results['flow']}</td>
145 HTML
146   end
147
148   matrix += (line == benchResults.size - 1 ? "<td></td>" : "") + '<td class="
149     unite flow">MB/s</td></tr><tr>'
150
151   column = 0
152   recvers.each do |recver, results|
153     matrix += '<td class="empty"></td>' if line == column
154     column += 1
155     # Mise en place des latences dans la matrice
156     matrix += <<-HTML
157     <td class="latency#{
158       case results['latency']
159         when latencys.first[:latency]
160           " min"
161         when latencys.last[:latency]
162           " max"
163         else
164           case latencys[latencys.index { |x| x[:sender] == sender && x[:
165             recver] == recver }][:ecc]
166             when 0..q1Latency
167               " low"
168             when q3Latency..latencys.last[:ecc]
169               " high"
170           end
171         end
172       }">#{results['latency']}</td>
173 HTML
174   end
175
176   matrix += (line == benchResults.size - 1 ? '<td class="empty"></td>' : "") + '<
177     td class="unite latency">&micro;s</td></tr>'
178   line += 1
179 end
180 # Ecriture du fichier de sortie html, avec la CSS et la matrix prealablement

```

```

remplie
177 File.open(html,'w') do |fo|
178   fo << <<-HTML
179     <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
        xhtml1/DTD/xhtml1-strict.dtd">
180     <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
181     <head>
182     <title>Flow Latency Tests - Matrix</title>
183     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
184     <style type="text/css">
185     <!--
186       body {
187         background-color: #EEE;
188         font-family: sans-serif;
189       }
190
191       h1, h2 {
192         text-align: center;
193         font-size: 30px;
194         margin: 5px;
195       }
196
197       h2 { font-size: 20px}
198
199       table {
200         border: 1px solid #000;
201         border-bottom: 0;
202         text-align: center;
203         margin: 30px auto;
204       }
205
206       th {
207         background-color: #BBB;
208         padding: 5px;
209       }
210
211       th.bot { border-bottom: 1px solid #000 }
212       th.right { border-right: 1px solid #000 }
213
214       td {
215         background-color: #AAA;
216         padding: 10px;
217       }
218
219       td.latency {
220         background-color: #EEE;
221         border-bottom: 1px solid #000;
222       }
223
224       td.flow { background-color: #CCC }
225
226       td.min, td.max {
227         background-color: red;
228         font-weight: bold;
229         color: white;
230       }
231
232       td.max { background-color: green }
233
234       td.high { color: green }
235       td.low { color: red }
236
237       td.empty { border-bottom: 1px solid #000 }
238       td.unite { font-weight: bold }
239
240       address {
241         text-align: center;
242         font-size: 10px;
243       }
244     -->
245   </style>
246 </head>
247

```

```
248 <body>
249 <h1>Latency flow tests</h1>
250 <h2>(with nice colors)</h2>
251
252 <table cellpadding="0" cellspacing="0">
253   #{matrix}
254 </table>
255
256 <address>&copy; ASRALL 2010 - All rights reserved (watch it pay)</address>
257 </body>
258 </html>
259
260 HTML
261 end
```

10 Tutoriel : Support de Infiniband dans un environnement personnalisé

Support d'Infiniband sur Grid5000

Modules Infiniband

Tests de performance

Enregistrement d'un environnement compatible

Licence ASRALL Nancy-Charlemagne
(Sébastien BADIA <seb@sebian.fr> <badia.seb@gmail.com> - Projet tutoré 2009-2010)

24 mars 2010

Table des matières

1	Déploiement	2
1.1	Upgrade Debian Lenny vers Testing	2
1.2	Erreur de udev avec le noyau	2
1.3	Drivers Infiniband	2
1.4	Personnalisation	3
2	Tests de reconnaissance	3
2.1	Ping Over Ib	3
2.2	RDMA	3
2.2.1	Perftests et Ibutils	3

1 Déploiement

1.1 Upgrade Debian Lenny vers Testing

L'upgrade du système n'est pas obligatoire pour installer les drivers. Cependant certains outils ne sont pas disponibles dans Lenny, ou sont trop anciens...

L'upgrade est une mise à jour classique, pour ce tuto nous utiliserons un environnement *lenny-x64-base*, disponible dans la base des environnements de Grid5000.

Passage en *testing* :

Commencer par remplacer les occurrences de **lenny** dans le fichier */etc/apt/sources.list* par **testing**

```
1 node: apt-get update && apt-get dist-upgrade
2 node: apt-get install firmware-bnx2 firmware-linux-nonfree firmware-linux firmware-
  linux-free
```

Note : Vous pouvez répondre par défaut à toutes les questions lors du processus de Màj, l'installation de "firmware-linux-nonfree" règle des soucis de modules avec le passage au noyau 2.6.32

1.2 Erreur de udev avec le noyau

Si lors du `dist-upgrade`, cette erreur survient :

```
1 Errors were encountered while processing:
2 /var/cache/apt/archives/udev_150-2_amd64.deb
```

Vous pouvez alors exécuter cette commande :

```
1 node: dpkg --configure -a
```

Cela permet de configurer tous les paquets restés en attente suite à l'erreur de udev.

Afin de rebooter sur le noyau 2.6.32 et continuer la mise à jour, il enregistre son environnement (*tgz-g5k*), puis précise les nouveaux noyaux et initrd installés (2.6.32-trunk-amd64), dans le *.env*, ou le *.dsc*. Redéployer avec le nouveau *.env*, ou le *.dsc* et relancer la commande de `dist-upgrade` pour terminer les mises à jour (Cette opération de redéploiement est apparentée à un reboot sur un pc classique).

On peut maintenant enregistrer son environnement¹ sur le *frontend*, afin de conserver un système upgradé propre, le *.env* n'a plus besoin d'être modifié.

1.3 Drivers Infiniband

Infiniband est entièrement intégré et utilisable sur les noyaux récent de linux, il vous faut donc juste activer certains modules.

Deux options s'offrent alors à vous, soit vous activer le module avec un *modprobe* après chaque démarrage du système.

Soit vous ajouter ces modules dans le fichier */etc/modules*, ceux-ci seront ainsi chargé au démarrage du système.

- `mlx4_ib` Module supportant la plus part des cartes Mellanox Infiniband
- `rdma_ucm`
- `ib_umad`
- `ib_uverbs`
- `ib_ipoib` Module pour le support de ip over infiniband
- `ib_srp`
- `bnx2`

```
1 node: modprobe mlx4_ib && modprobe rdma_ucm && modprobe ib_umad
2 node: modprobe ib_uverbs && modprobe ib_ipoib && modprobe ib_srp && modprobe bnx2
```

1. Réf : https://www.grid5000.fr/mediawiki/index.php/Deploy_environment (section **Create a new environment from a customized environment**)

1.4 Personnalisation

Changer le mot de passe root (default = grid5000) :

```
1 node: passwd
```

Créer votre utilisateur² (exécuter la commande `id` sur un *frontend* afin de récupérer les informations à fournir ; Le nom du groupe correspond théoriquement à la ville d'inscription) :

```
1 node: addgroup --gid <GID> <NOM_GROUPE>
2 node: adduser --uid <UID> --ingroup <NOM_GROUPE> <LOGIN>
```

Enregistrer l'environnement³.

2 Tests de reconnaissance

```
1 node: ibv_devinfo
2 node: ibstat
```

2.1 Ping Over Ib

Communication entre deux machines (Ping Over Ib)

Sur la première machine on va déterminer le guid de la carte pour que la deuxième machine puisse "ping" :

```
1 node: ibstat -p
```

Sur la seconde :

```
1 node: ibping -G <guid-hote-vise>
```

2.2 RDMA

Remote Direct Memory Access sur Infiniband (Bande Passante)

Sur la première machine :

```
1 node: ib_rdma_bw
```

Sur la seconde :

```
1 node: ib_rdma_bw <Node1>
```

2.2.1 Perftests et Ibutils

Les Paquets Perftest et Ibutils, apportent un panel de commandes de tests supplémentaires, les classiques test de ping-pong mais aussi bien d'autres.

2. Réf : https://www.grid5000.fr/mediawiki/index.php/Deploy_environment (section **Customization**)

3. Réf : https://www.grid5000.fr/mediawiki/index.php/Deploy_environment (section **Create a new environment from a customized environment**)

11 Tutoriel : Support de Myrinet dans un environnement personnalisé

Drivers Myrinet pour Grid5000

Compilation des drivers

Construction des paquets Debian

Enregistrement d'un environnement compatible

Licence A.S.R.A.L.L. Nancy-Charlemagne
(Julien VAUBOURG <julien@vaubourg.com> - Projet tutoré 2009-2010)

13 février 2010

Table des matières

1	Upgrader son environnement	2
1.1	Upgrade	2
1.2	Erreur habituelle	2
2	Recompilation du noyau et construction d'un paquet	2
3	Compilation des drivers et construction d'un paquet	3
3.1	Compiler le driver dans l'optique de faire le paquet	3
3.2	Préparation du paquet	3
3.2.1	Copie des fichiers	3
3.2.2	Fichier de version du paquet	3
3.2.3	Service de lancement des drivers	4
3.2.4	Lancement automatique du service	5
3.3	Construction et sauvegarde du paquet	5
4	Enregistrement d'un environnement Myrinet	5
4.1	Installation des drivers	5
4.2	Customisations utiles	5
5	Tests	6
5.1	Reconnaissance de la carte	6
5.2	Communication entre deux machines	6
6	Références	6

1 Upgrader son environnement

1.1 Upgrade

L'upgrade du système n'est pas obligatoire pour installer les drivers. Toutefois, une fois les différentes compilations effectuées, il ne sera plus temps d'upgrader le système, au risque de devoir tout recompiler.

L'upgrade se fait à partir d'un environnement *lenny-x64-base* (au moment de ce tuto), disponible dans la base des environnements.

Passage en *testing* :

```
1 node: sed -i "s/lenny/testing/" /etc/apt/sources.list
2 node: apt-get update
3 node: apt-get -y dist-upgrade => reponse par default a toutes les questions.
4 node: apt-get install 'firmware-bnx2*'
```

Enregistrer son environnement¹ sur le *frontend*, afin de conserver un système upgradé propre. La suite de ce tuto (construction des paquets) pourra se faire sans redéployer.

1.2 Erreur habituelle

Si lors du `dist-upgrade`, cette erreur survient :

```
1 Errors were encountered while processing:
2 /var/cache/apt/archives/udev_150-2_amd64.deb
```

Exécuter cette commande :

```
1 node: dpkg --configure -a => tout valider.
```

Puis exécuter la commande qui suit le `dist-upgrade` dans la procédure plus haut, puis enregistrer son environnement en précisant les nouveaux noyaux et initrd installés, dans le `.env`. Redéployer et relancer la commande de `dist-upgrade` pour terminer les mises à jour (un simple reboot aurait pu suffire, mais ça ne semble pas possible).

2 Recompilation du noyau et construction d'un paquet

Les drivers Myrinet doivent impérativement être compilés avec le même compilateur que le noyau. Le seul moyen de s'en assurer est de recompiler le noyau (2.6.32 au moment de ce tuto).

```
1 node: apt-get install linux-source-<KVERSION>
2 node: cd /usr/src
3 node: tar xvfj linux-source-<KVERSION>.tar.bz2
4 node: cd linux-source-<KVERSION>/
5 node: cp /boot/config-<KVERSION_ANTIEN> .config
6 node: make menuconfig => exit puis yes.
7 node: make-kpkg --revision=<KVERSION> --append-to-version=myrinet kernel-image
8 node: mv linux-image-<KVERSION>myrinet{_,_}amd64.deb => juste plus joli.
9 node: scp ../linux-image-<KVERSION>myrinet_amd64.deb <LOGIN>@frontend: =>
sauvegarde.
```

¹Réf : https://www.grid5000.fr/mediawiki/index.php/Deploy_environment (section Create a new environment from a customized environment)

3 Compilation des drivers et construction d'un paquet

Télécharger les drivers² (version actuelle : `mx_1.2.11.tar.gz`). Pour obtenir le login et le mot de passe, remplir le formulaire dédié³.

Attention : le driver `Myri10GE_Linux_1.5.14` permet uniquement de simuler de l'ethernet classique sur une carte Myrinet (la transformant ainsi en une vulgaire carte ethernet 10G, au mieux). C'est donc bien le driver `mx_*` qu'il faut récupérer.

3.1 Compiler le driver dans l'optique de faire le paquet

```
1 node: tar xvzf mx_<MXVERSION>.tar.gz
2 node: cd mx-<MXVERSION>
3 node: ./configure --with-linux=/usr/src/linux-source-<KVERSION> --enable-kernel-lib
4 node: make && make install
```

Le driver s'est installé dans `/opt/mx`. ces fichiers serviront uniquement à connaître la liste précise des fichiers qui lui sont associés. Recompiler le driver en installant les fichiers à l'emplacement souhaité (dans `/usr/local`).

```
1 node: ./configure --with-linux=/usr/src/linux-source-<KVERSION> --enable-kernel-lib
  --prefix /usr/local
2 node: make && make install
```

3.2 Préparation du paquet

3.2.1 Copie des fichiers

```
1 node: DEB_MYRINET=/tmp/mx-<MXVERSION>_<KVERSION>myrinet_amd64
2 node: mkdir -p $DEB_MYRINET/usr/local
3 node: for i in $(find /opt/mx/* | sed "s/~/opt/mx/"); do [ -d "/usr/local/$i"
  ] && mkdir -p $DEB_MYRINET/usr/local/$i || cp /usr/local/$i $DEB_MYRINET/usr/
  local/$i; done
4 node: mkdir $DEB_MYRINET/DEBIAN
```

3.2.2 Fichier de version du paquet

```
1 node: vim $DEB_MYRINET/DEBIAN/control
```

²<http://www.myri.com/scs/download-mx10g.html>

³<http://www.myri.com/scs/loginrequest.php> (si vous n'êtes pas assez convaincant, le support vous demandera un numéro de série : « Please send a serial number from one component - the 6-digit number prefixed by "SN" on the white label - so that we can verify your license »)

⁴<http://www.myri.com/scs/download-Myri10GE.html>

Ajouter :

```

1 Package: mx-<MXVERSION>-<KVERSION>myrinet
2 Version: 0.1
3 Section: base
4 Priority: optional
5 Architecture: amd64
6 Depends: linux-image-<KVERSION>myrinet
7 Maintainer: Julien VAUBOURG <julien@vaubourg.com>
8 Description: Drivers compiled with the same compiler than linux-image-<KVERSION>
   myrinet

```

3.2.3 Service de lancement des drivers

```

1 node: mkdir -p $DEB_MYRINET/etc/init.d
2 node: vim $DEB_MYRINET/DEBIAN/etc/init.d/myrinet

```

Ajouter :

```

1 #!/bin/sh
2
3 #### BEGIN INIT INFO
4 # Provides:          mx_local_install
5 # Required-Start:
6 # Required-Stop:
7 # Default-Start:    2 3 4 5
8 # Default-Stop:
9 # Short-Description: Launch and stop the myrinet drivers
10 #### END INIT INFO
11
12 case "$1" in
13     start)
14         /usr/local/sbin/mx_local_install
15         /etc/init.d/mx start
16         /sbin/ifconfig myri0 up
17         ;;
18
19     stop)
20         /sbin/ifconfig myri0 down
21         /etc/init.d/mx stop
22         ;;
23
24     restart)
25         /etc/init.d/myrinet stop
26         /etc/init.d/myrinet start
27         ;;
28
29     *)
30         echo "Usage: /etc/init.d/myrinet {start|stop|restart}"
31         exit 1
32 esac
33
34 exit 0

```

3.2.4 Lancement automatique du service

```
1 node: echo -e \#\!/ bin/sh \n/usr/sbin/update-rc.d \{myrinet\ defaults,-f\ mx\
  remove\} > $DEB_MYRINET/DEBIAN/postinst
2 node: chmod 755 $DEB_MYRINET/DEBIAN/postinst
3 node: chmod +x $DEB_MYRINET/etc/init.d/myrinet
```

3.3 Construction et sauvegarde du paquet

```
1 node: dpkg-deb -b $DEB_MYRINET
2 node: scp ${DEB_MYRINET}.deb <LOGIN>@frontend :
```

4 Enregistrement d'un environnement Myrinet

4.1 Installation des drivers

Depuis un environnement *lenny-x64-base* (au moment de ce tuto) éventuellement upgradé en *testing*⁵ :

```
1 node: dpkg -i linux-image-<KVERSION>myrinet_amd64.deb mx-<MXVERSION>_<KVERSION>
  myrinet_amd64.deb
2 node: mkinitramfs -o /boot/initrd.img-<KVERSION>myrinet <KVERSION>myrinet
```

4.2 Customisations utiles

Changer le mot de passe root :

```
1 node: passwd
```

Créer votre utilisateur⁶ (exécuter la commande `id` sur un *frontend* afin de récupérer les informations à fournir ; Le nom du groupe correspond théoriquement à la ville d'inscription) :

```
1 node: addgroup --gid <GID> <NOM GROUPE>
2 node: adduser --uid <UID> --ingroup <NOM GROUPE> <LOGIN>
```

Ajouter l'utilisateur aux sudoers :

```
1 node: echo "<LOGIN> ALL=(ALL) ALL" >> /etc/sudoers
```

Enregistrer l'environnement⁷ et le redéployer (en n'oubliant pas de changer le `initrd` et le noyau dans le `.env` par ceux taggués *myrinet*).

Les drivers seront automatiquement chargés.

⁵Voir la section **Upgrader son environnement**

⁶Réf : https://www.grid5000.fr/mediawiki/index.php/Deploy_environment (section **Customization**)

⁷Réf : https://www.grid5000.fr/mediawiki/index.php/Deploy_environment (section **Create a new environment from a customized environment**)

5 Tests

5.1 Reconnaissance de la carte

```
1 node: mx_myri_info
```

5.2 Communication entre deux machines

Sur la première machine :

```
1 node: mx_pingpong -s
```

Sur la seconde :

```
1 node: mx_pingpong -M 4 -E 1048577 -d <NODE>.<SITE>.grid5000.fr:0
```

Si `mx_pingpong` répond une douzaine de lignes de stats de performance, c'est que ça fonctionne.

6 Références

- Page Grid5000 dédiée à Myrinet :
<https://www.grid5000.fr/mediawiki/index.php/Myrinet>
- Page Grid5000 dédiée au déploiement et à l'enregistrement d'environnements :
https://www.grid5000.fr/mediawiki/index.php/Deploy_environment