Les clusters de calcul

Configuration et outils pour la recette

Tuteur : M. Lucas Nussbaum

S. Badia, G. Delourmel,
L. Didry & J. Vaubourg

IUT Nancy Charlemagne, Université Nancy 2

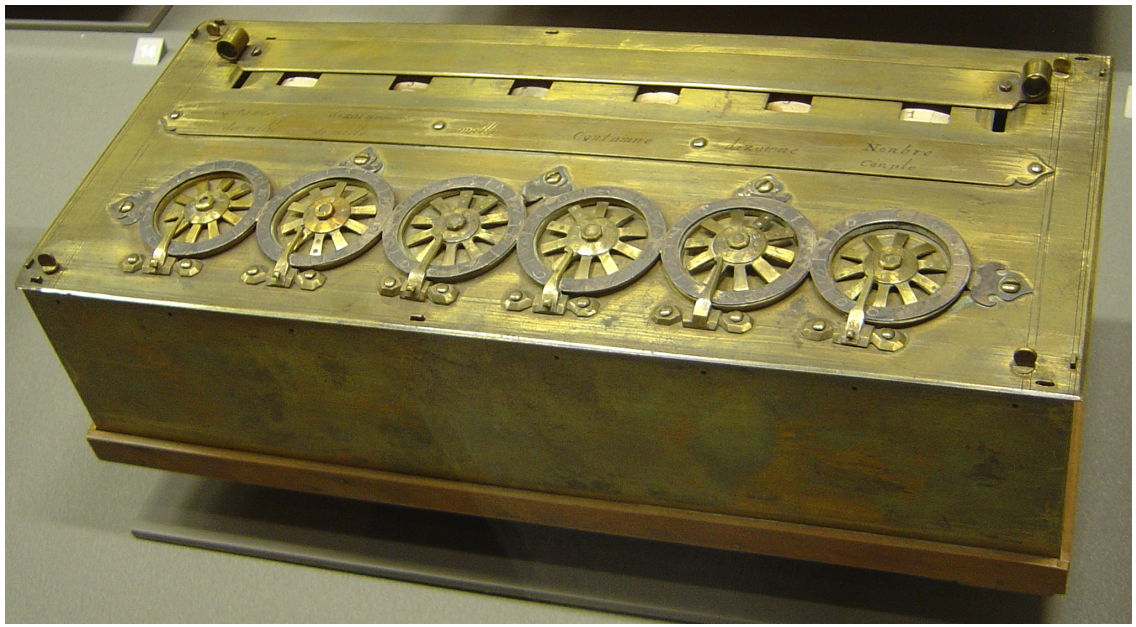
31 mars 2010

Partie I

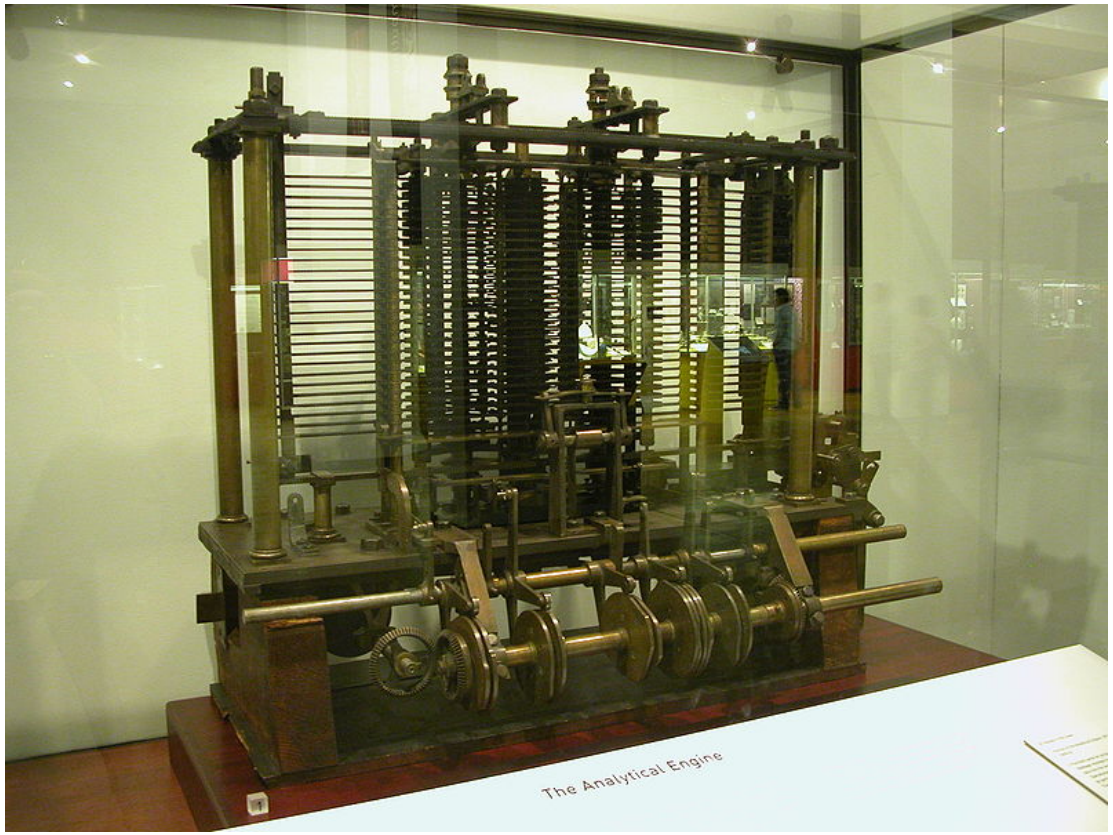
Introduction

Historique du calcul haute performance

XVII^{ème} siècle : la première machine mécanique, la *pascaline*

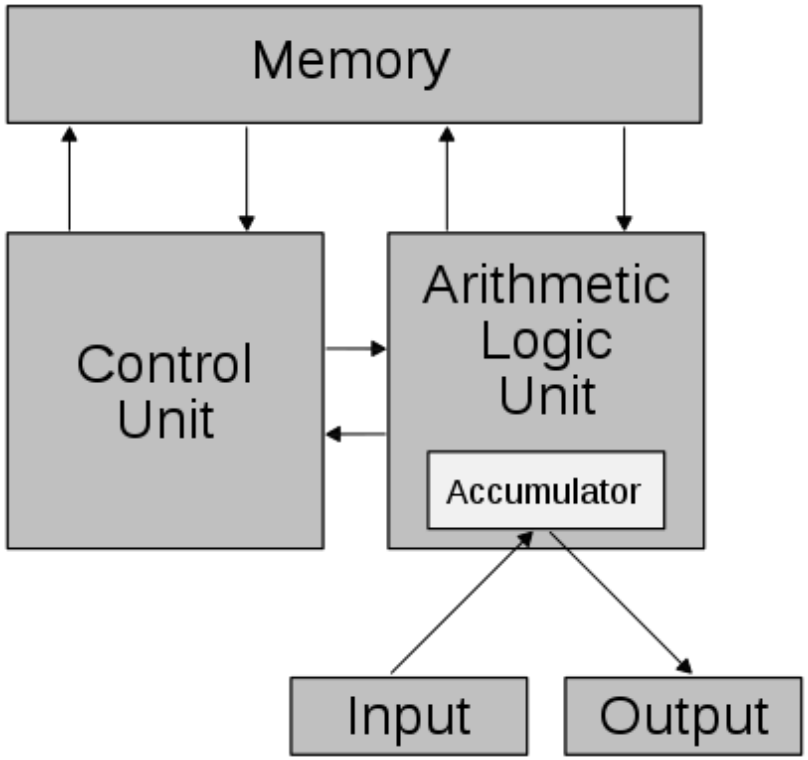


XIX$^{\text{ème}}$ siècle : la machine mécanique de Babbage



1945 : architecture de Von Neumann

1956 : premiers ordinateurs basés sur cette architecture et utilisant les possibilités des semi-conducteurs



1970 : la société *Cray* développe une architecture permettant un coût d'un million de dollars pour un million d'opérations par seconde

Historique du calcul haute performance

- Trente ans plus tard, le coût reste le même pour un nombre d'opérations multiplié par un million
- Aujourd'hui, le plus puissant supercalculateur possède un capacité de calcul de 1759 *téraflops* (10^{12} opérations par seconde)
- L'avenir : certains pensent pouvoir atteindre l'*exaflop* (10^{18} opérations par seconde) d'ici 2015

Historique du calcul haute performance

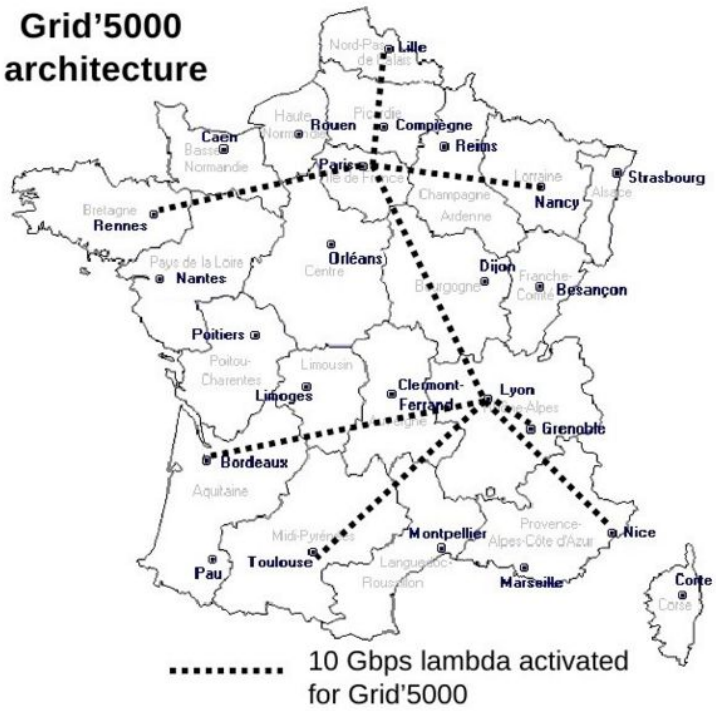
- Trente ans plus tard, le coût reste le même pour un nombre d'opérations multiplié par un million
- Aujourd'hui, le plus puissant supercalculateur possède un capacité de calcul de 1759 *téraflops* (10^{12} opérations par seconde)
- L'avenir : certains pensent pouvoir atteindre l'*exaflop* (10^{18} opérations par seconde) d'ici 2015

Historique du calcul haute performance

- Trente ans plus tard, le coût reste le même pour un nombre d'opérations multiplié par un million
- Aujourd'hui, le plus puissant supercalculateur possède un capacité de calcul de 1759 *téraflops* (10^{12} opérations par seconde)
- L'avenir : certains pensent pouvoir atteindre l'*exaflop* (10^{18} opérations par seconde) d'ici 2015

Organisation du projet Grid5000

Le but du projet Grid5000 est d'obtenir une grille de calcul de 5000 processeurs répartis sur la France entière en 9 sites reliés via *RENATER*



Organisation du projet Grid5000

- Chaque site possède un ou plusieurs *clusters*
- Chaque site est géré par son (ses) propre(s) administrateur(s)
- Cela peut conduire à un comportement différent selon le site :
  - les logiciels ou bibliothèques installés ne sont pas forcément les mêmes
  - certains permettent un accès depuis l'extérieur, comme Lyon, mais pas d'autres, comme Nancy

Organisation du projet Grid5000

- Chaque site possède un ou plusieurs *clusters*
- Chaque site est géré par son (ses) propre(s) administrateur(s)
- Cela peut conduire à un comportement différent selon le site :
  - les logiciels ou bibliothèques installés ne sont pas forcément les mêmes
  - certains permettent un accès depuis l'extérieur, comme Lyon, mais pas d'autres, comme Nancy

Organisation du projet Grid5000

- Chaque site possède un ou plusieurs *clusters*
- Chaque site est géré par son (ses) propre(s) administrateur(s)
- Cela peut conduire à un comportement différent selon le site :
  - les logiciels ou bibliothèques installés ne sont pas forcément les mêmes
  - certains permettent un accès depuis l'extérieur, comme Lyon, mais pas d'autres, comme Nancy

Organisation du projet Grid5000

- Chaque site possède un ou plusieurs *clusters*
- Chaque site est géré par son (ses) propre(s) administrateur(s)
- Cela peut conduire à un comportement différent selon le site :
  - les logiciels ou bibliothèques installés ne sont pas forcément les mêmes
  - certains permettent un accès depuis l'extérieur, comme Lyon, mais pas d'autres, comme Nancy

Organisation du projet Grid5000

- Chaque site possède un ou plusieurs *clusters*
- Chaque site est géré par son (ses) propre(s) administrateur(s)
- Cela peut conduire à un comportement différent selon le site :
  - les logiciels ou bibliothèques installés ne sont pas forcément les mêmes
  - certains permettent un accès depuis l'extérieur, comme Lyon, mais pas d'autres, comme Nancy

Utilisation

- On se connecte par *SSH* à un site le permettant
- Immédiatement, le répertoire `/home` de l'utilisateur est monté en *NFS*
- On réserve un certain nombre de nœuds pour un temps donné
- Cette réservation peut être précise quant au nombre de machines et à leurs caractéristiques et peut se faire à l'avance

Utilisation

- On se connecte par *SSH* à un site le permettant
- Immédiatement, le répertoire `/home` de l'utilisateur est monté en *NFS*
- On réserve un certain nombre de nœuds pour un temps donné
- Cette réservation peut être précise quant au nombre de machines et à leurs caractéristiques et peut se faire à l'avance

Utilisation

- On se connecte par *SSH* à un site le permettant
- Immédiatement, le répertoire `/home` de l'utilisateur est monté en *NFS*
- On réserve un certain nombre de nœuds pour un temps donné
- Cette réservation peut être précise quant au nombre de machines et à leurs caractéristiques et peut se faire à l'avance

Utilisation

- On se connecte par *SSH* à un site le permettant
- Immédiatement, le répertoire `/home` de l'utilisateur est monté en *NFS*
- On réserve un certain nombre de nœuds pour un temps donné
- Cette réservation peut être précise quant au nombre de machines et à leurs caractéristiques et peut se faire à l'avance

Objectifs du projet

Développer une suite d'outils permettant de vérifier :

- la conformité des *clusters* à l'appel d'offre auquel il fait suite :
  - conformité matérielle
  - conformité des vitesses des disques durs
- les performances réseau
- la bonne interconnexion des nœuds

Objectifs du projet

Développer une suite d'outils permettant de vérifier :

- la conformité des *clusters* à l'appel d'offre auquel il fait suite :
  - conformité matérielle
  - conformité des vitesses des disques durs
- les performances réseau
- la bonne interconnexion des nœuds

Objectifs du projet

Développer une suite d'outils permettant de vérifier :

- la conformité des *clusters* à l'appel d'offre auquel il fait suite :
  - conformité matérielle
  - conformité des vitesses des disques durs
- les performances réseau
- la bonne interconnexion des nœuds

Objectifs du projet

Développer une suite d'outils permettant de vérifier :

- la conformité des *clusters* à l'appel d'offre auquel il fait suite :
  - conformité matérielle
  - conformité des vitesses des disques durs
- les performances réseau
- la bonne interconnexion des nœuds

Objectifs du projet

Développer une suite d'outils permettant de vérifier :

- la conformité des *clusters* à l'appel d'offre auquel il fait suite :
  - conformité matérielle
  - conformité des vitesses des disques durs
- les performances réseau
- la bonne interconnexion des nœuds

Partie II

Inventaire

Problèmatique

- Vérifier la conformité des nœuds à l'appel d'offre
- Vérifier l'homogénéité par *cluster*
- Vérifier la conformité à l'*OAR Database*
- Détecter des pannes ou des disfonctionnements éventuels

Problèmatique

- Vérifier la conformité des nœuds à l'appel d'offre
- Vérifier l'homogénéité par *cluster*
- Vérifier la conformité à l'*OAR Database*
- Détecter des pannes ou des disfonctionnements éventuels

Problèmatique

- Vérifier la conformité des nœuds à l'appel d'offre
- Vérifier l'homogénéité par *cluster*
- Vérifier la conformité à l'*OAR Database*
- Détecter des pannes ou des disfonctionnements éventuels

Problèmatique

- Vérifier la conformité des nœuds à l'appel d'offre
- Vérifier l'homogénéité par *cluster*
- Vérifier la conformité à l'*OAR Database*
- Détecter des pannes ou des disfonctionnements éventuels

Outils utilisés

- Langage *Perl*
- Différents outils *Unix* :
- Différents modules *Perl* :

Outils utilisés

- Langage *Perl*
- Différents outils *Unix* :
- Différents modules *Perl* :

Outils utilisés

- Langage *Perl*
- Différents outils *Unix* :

```
lshw
lspci
uname
ifconfig
cat /proc/cpuinfo
```

- Différents modules *Perl* :

Outils utilisés

- Langage *Perl*
- Différents outils *Unix* :

```
lshw
lspci
uname
ifconfig
cat /proc/cpuinfo
```

- Différents modules *Perl* :

Outils utilisés

- Langage *Perl*
- Différents outils *Unix* :

```
lshw
lspci
uname
ifconfig
cat /proc/cpuinfo
```

- Différents modules *Perl* :

Outils utilisés

- Langage *Perl*
- Différents outils *Unix* :

```
lshw
lspci
uname
ifconfig
cat /proc/cpuinfo
```

- Différents modules *Perl* :

Outils utilisés

- Langage *Perl*
- Différents outils *Unix* :

```
lshw
lspci
uname
ifconfig
cat /proc/cpuinfo
```

- Différents modules *Perl* :

Outils utilisés

- Langage *Perl*
- Différents outils *Unix* :
- Différents modules *Perl* :

Outils utilisés

- Langage *Perl*
- Différents outils *Unix* :
- Différents modules *Perl* :

```
YAML
Data::Dumper
XML::Simple
POSIX ":sys_wait_h"
```

Outils utilisés

- Langage *Perl*
- Différents outils *Unix* :
- Différents modules *Perl* :

```
YAML
Data::Dumper
XML::Simple
POSIX ":sys_wait_h"
```

Outils utilisés

- Langage *Perl*
- Différents outils *Unix* :
- Différents modules *Perl* :

```
YAML
Data::Dumper
XML::Simple
POSIX ":sys_wait_h"
```

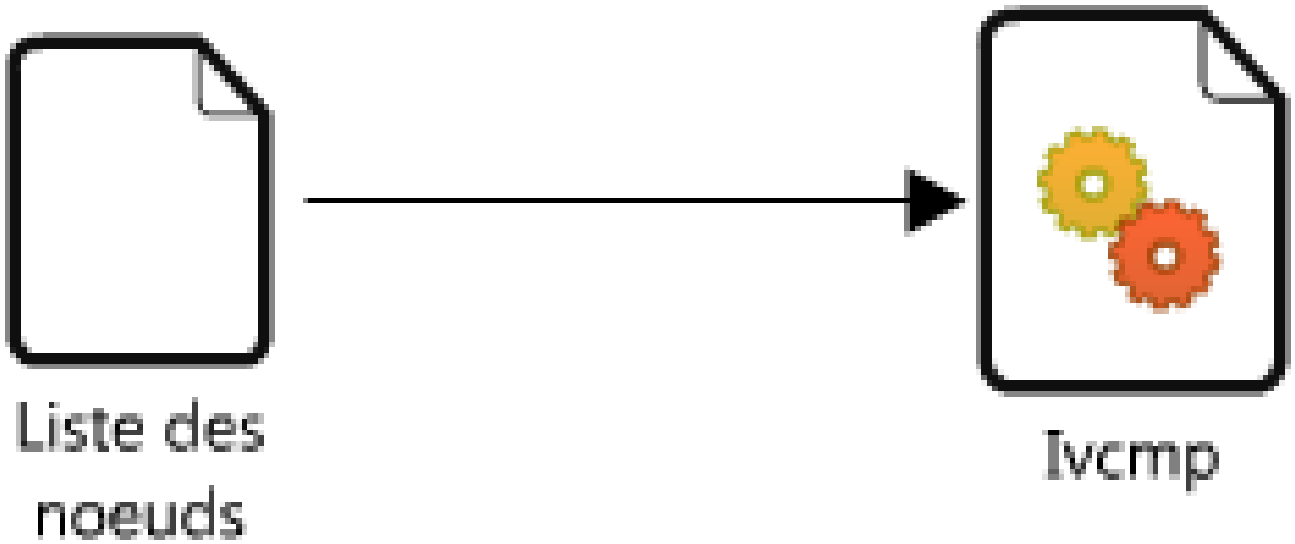
Outils utilisés

- Langage *Perl*
- Différents outils *Unix* :
- Différents modules *Perl* :

```
YAML
Data::Dumper
XML::Simple
POSIX ":sys_wait_h"
```

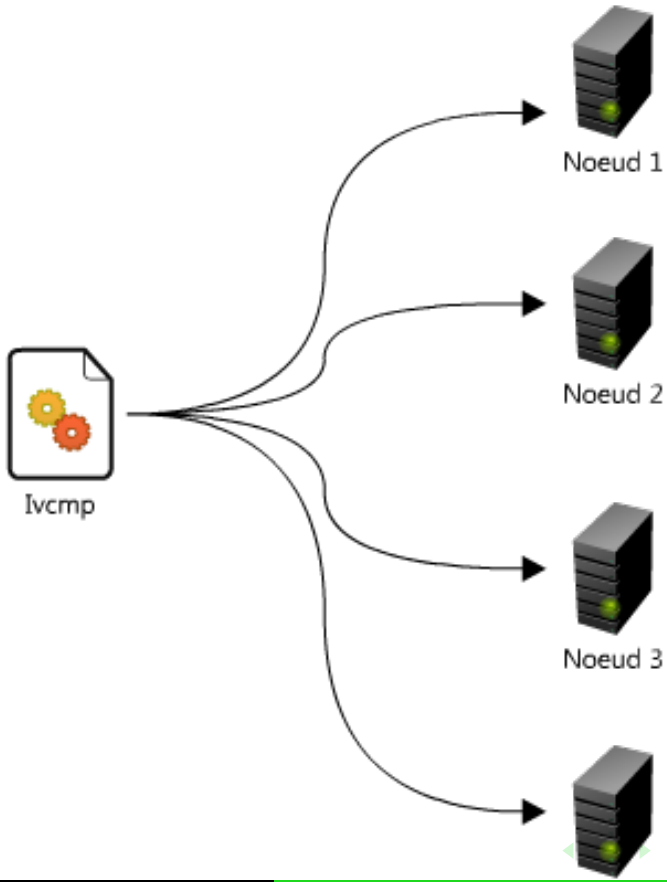
Algorithme

- Récupération de la liste des nœuds



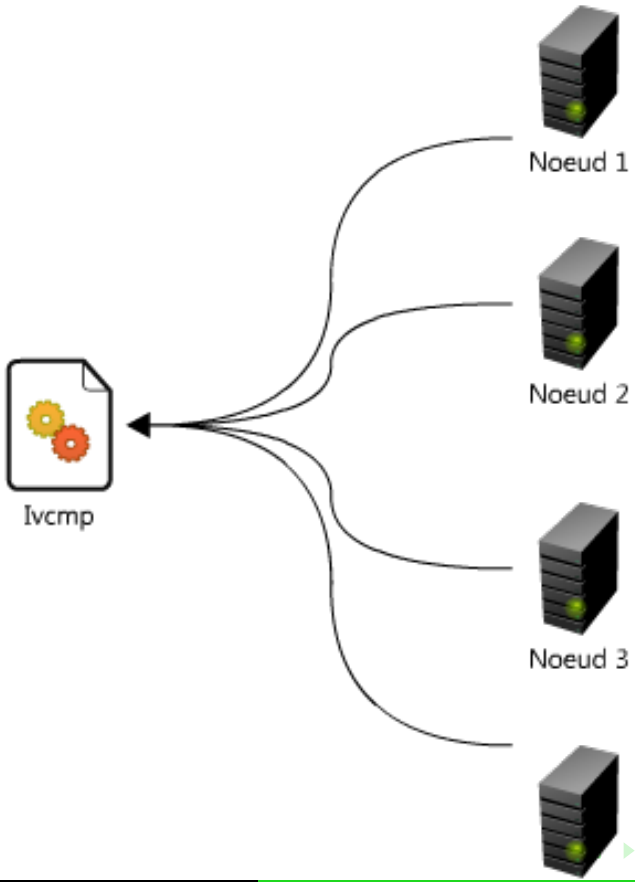
Algorithme

- Installation de `lshw` en parallèle



Algorithme

- Récupération des informations en parallèle

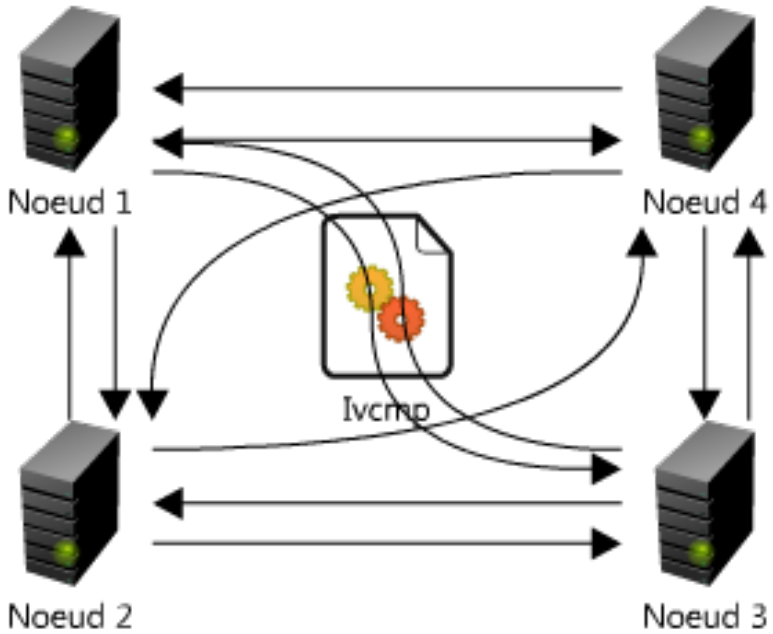


Algorithme

- Récupération des informations de l'*OAR Database* et comparaison avec les nœuds

Algorithme

- Comparaison des nœuds entre eux



Algorithme

- Sortie au format *YAML*

```
--- Notice that the printed value are from Unix tools, not from OAR database ---
1 :
  cluster : griffon
  cpuarch : x86_64
  cpucore : 4
  cpufreq : 2.5
  cputype : Intel(R) Xeon(R) CPU L5420 @ 2.50GHz
  diskmodel : Hitachi HDP72503
  disksize : 320
  disktype : sata
  ib10g : ~
  ib10gmodel : ~
  ib20g : 1
  ib20gmodel : MT26418
  memnode : 16384
  myri10g : ~
  myri10gmodel : ~
  myri2g : ~
  myri2gmodel : ~
  nodes :
    griffon-90.nancy.grid5000.fr : ~
    griffon-92.nancy.grid5000.fr : ~
```

Algorithme

- Affichage des incohérences au niveau de l'*OAR Database*

```
----- The Oar database cpufreq information of the node grelon-53.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-53.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-54.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-54.nancy.grid5000.fr is not the same as we check on the node -----
      OAR database : 2          Unix tools : 1
```

Options disponibles

```
Ivcmp - copyright (c) 2010 Guillaume Delourmel <delourmel_guillaume@yahoo.fr>, Luc Didry <lucdidry@free.fr>

Inventory script which compares the informations in the Oar database
and informations returned by Unix tools such as lshw, lspci, etc.
It can be used to see if all the nodes of a cluster have the same material

You need a ssh key with an empty passphrase in order to avoid to type the passphrase for each node
You need to be root on the nodes in order to get all the required informations with lshw

Usage : ./ivcmp [OPTIONS]...

    -f, --file [FILE]                       specify the file containing the list of the nodes ($OAR_FILE_NODES by default)
                                            This file need to have one node per line (duplicate nodes will be ignored)
    -o, --output [FILE]                     print the result in the file in YAML format
    -p, --parameter CRITERION [CRITERION]...  check the nodes against specified criteria
    -i, --install                           automatically install lshw on nodes (lshw is required)
    -r, --remove                            automatically remove the directories containing the ouput of the different used tools
    -h, --help                              print this help and exit
```

Quelques incohérences relevées

Lyon 3 nœuds possèdent deux disques durs de 73Go au lieu d'un seul

Nancy un nœud possède une carte Infiniband différente du reste du *cluster*

Orsay un grand nombre de nœuds du *Gdx* possède une fréquence de CPU différente

Sophia les nœuds du *cluster Sol* n'ont qu'une seule carte ethernet activée contre deux annoncées dans l'*OAR database*

...

Objectifs

Le but de ce script était de vérifier :

- La vitesse d'écriture
- La vitesse de lecture
- Le nombre d'opérations

Objectifs

Le but de ce script était de vérifier :

- La vitesse d'écriture
- La vitesse de lecture
- Le nombre d'opérations

Objectifs

Le but de ce script était de vérifier :

- La vitesse d'écriture
- La vitesse de lecture
- Le nombre d'opérations

Outils utilisés

- Langage *Perl*
- Outils *Unix*
  - `free` : pour récupérer la taille de la mémoire ram (nécessaire pour `bonnie`)
  - `ssh` : pour faire les tests et l'installation de `bonnie` sur les nœuds
  - `Bonnie++` : pour effectuer les tests sur les disques durs. on utilise le double de la taille de la mémoire ram pour le test. Par exemple pour 16Go de ram on aura :

```
bonnie++ -s 32768 -d /tmp -m griffon-92.nancy.grid5000.fr -u root
```

Outils utilisés

- Langage *Perl*
- Outils *Unix*
  - `free` : pour récupérer la taille de la mémoire ram (nécessaire pour `bonnie`)
  - `ssh` : pour faire les tests et l'installation de `bonnie` sur les nœuds
  - `Bonnie++` : pour effectuer les tests sur les disques durs. on utilise le double de la taille de la mémoire ram pour le test. Par exemple pour 16Go de ram on aura :

```
bonnie++ -s 32768 -d /tmp -m griffon-92.nancy.grid5000.fr -u root
```

Outils utilisés

- Langage *Perl*
- Outils *Unix*
  - `free` : pour récupérer la taille de la mémoire ram (nécessaire pour `bonnie`)
  - `ssh` : pour faire les tests et l'installation de `bonnie` sur les nœuds
  - `Bonnie++` : pour effectuer les tests sur les disques durs. on utilise le double de la taille de la mémoire ram pour le test. Par exemple pour 16Go de ram on aura :

```
bonnie++ -s 32768 -d /tmp -m griffon-92.nancy.grid5000.fr -u root
```

Outils utilisés

- Langage *Perl*
- Outils *Unix*
  - `free` : pour récupérer la taille de la mémoire ram (nécessaire pour `bonnie`)
  - `ssh` : pour faire les tests et l'installation de `bonnie` sur les nœuds
  - `Bonnie++` : pour effectuer les tests sur les disques durs. on utilise le double de la taille de la mémoire ram pour le test. Par exemple pour 16Go de ram on aura :

```
bonnie++ -s 32768 -d /tmp -m griffon-92.nancy.grid5000.fr -u root
```

Outils utilisés

- Langage *Perl*
- Outils *Unix*
  - `free` : pour récupérer la taille de la mémoire ram (nécessaire pour `bonnie`)
  - `ssh` : pour faire les tests et l'installation de `bonnie` sur les nœuds
  - `Bonnie++` : pour effectuer les tests sur les disques durs. on utilise le double de la taille de la mémoire ram pour le test. Par exemple pour 16Go de ram on aura :

    ```
    bonnie++ -s 32768 -d /tmp -m griffon-92.nancy.grid5000.fr -u root
    ```

Fonctionnement

Algorithme :

- Récupération de la liste des nœuds réservés
- Création de processus fils (un par nœud)
- Chaque fils exécute les actions suivantes sur son nœud avec `ssh` :
  - il installe Bonnie++
  - il récupère la taille de la mémoire ram
  - il exécute la commande `bonnie`
- Le père attend la fin des processus fils et formate les données comme l'a demandé l'utilisateur

Fonctionnement

Algorithme :

- Récupération de la liste des nœuds réservés
- Création de processus fils (un par nœud)
- Chaque fils exécute les actions suivantes sur son nœud avec `ssh` :
  - il installe Bonnie++
  - il récupère la taille de la mémoire ram
  - il exécute la commande `bonnie`
- Le père attend la fin des processus fils et formate les données comme l'a demandé l'utilisateur

Fonctionnement

Algorithme :

- Récupération de la liste des nœuds réservés
- Création de processus fils (un par nœud)
- Chaque fils exécute les actions suivantes sur son nœud avec `ssh` :
  - il installe Bonnie++
  - il récupère la taille de la mémoire ram
  - il exécute la commande `bonnie`
- Le père attend la fin des processus fils et formate les données comme l'a demandé l'utilisateur

Fonctionnement

Algorithme :

- Récupération de la liste des nœuds réservés
- Création de processus fils (un par nœud)
- Chaque fils exécute les actions suivantes sur son nœud avec `ssh` :
  - il installe Bonnie++
  - il récupère la taille de la mémoire ram
  - il exécute la commande `bonnie`
- Le père attend la fin des processus fils et formate les données comme l'a demandé l'utilisateur

Fonctionnement

Algorithme :

- Récupération de la liste des nœuds réservés
- Création de processus fils (un par nœud)
- Chaque fils exécute les actions suivantes sur son nœud avec `ssh` :
  - il installe Bonnie++
  - il récupère la taille de la mémoire ram
  - il exécute la commande `bonnie`
- Le père attend la fin des processus fils et formate les données comme l'a demandé l'utilisateur

Fonctionnement

Algorithme :

- Récupération de la liste des nœuds réservés
- Création de processus fils (un par nœud)
- Chaque fils exécute les actions suivantes sur son nœud avec `ssh` :
  - il installe Bonnie++
  - il récupère la taille de la mémoire ram
  - il exécute la commande `bonnie`
- Le père attend la fin des processus fils et formate les données comme l'a demandé l'utilisateur

Fonctionnement

Algorithme :

- Récupération de la liste des nœuds réservés
- Création de processus fils (un par nœud)
- Chaque fils exécute les actions suivantes sur son nœud avec `ssh` :
  - il installe Bonnie++
  - il récupère la taille de la mémoire ram
  - il exécute la commande `bonnie`
- Le père attend la fin des processus fils et formate les données comme l'a demandé l'utilisateur

Utilisation

- Utilisation du script ivbon :

```
Ivbon - copyright (c) 2010 Guillaume Delourmel <delourmel_guillaume@yahoo.fr>

Script to test all the hard disk drive of the nodes on a cluster
Usage : ivbon [OPTIONS]...
    -f, --file [FILE]                   file which contains the list of node to test
    -i, --install                       automatically install bonnie++ on nodes (bonnie++ is required)
    -y, --yaml                          output test as yaml
    -r, --remove                        remove all temporary files create by the script
    -o, --output [FILE]                 print the result in the file in YAML format, default is bonnie.txt
    -h, --help                          print this help and exit
```

Utilisation

- Exemple de sortie au format texte :

```
Average_speed_of_writing: 60392.3
Average_speed_of_reading: 86875.3
Average_number_of_operations: 167.1
                Node                    |Writing        |Reading        |Number_of_operations
griffon-92.nancy.grid5000.fr            |44509          |77547          |161.0
griffon-90.nancy.grid5000.fr            |45851          |76340          |167.6
griffon-91.nancy.grid5000.fr            |90817          |106739         |172.6
```

Utilisation

- Exemple de sortie au format YAML :

```
Average_speed_of_writing: 55510
Average_speed_of_reading: 67703.7
Average_number_of_operations: 195
grelon-88.nancy.grid5000.fr
  writing: 54096
  reading: 66904
  number_of_operations: 195.0
grelon-89.nancy.grid5000.fr
  writing: 56154
  reading: 66971
  number_of_operations: 193.3
grelon-87.nancy.grid5000.fr
  writing: 56280
  reading: 69236
  number_of_operations: 196.8
```

Problèmes rencontrés

- Changement du format de sortie de l'outil `oarnodes` à trois reprises
- Parallélisation complète du script Ivcmp impossible
- Bug du routeur *RENATER*
- Modules *Perl* nécessaires pas toujours disponibles sur tous les sites

Problèmes rencontrés

- Changement du format de sortie de l'outil `oarnodes` à trois reprises
- Parallélisation complète du script Ivcmp impossible
- Bug du routeur *RENATER*
- Modules *Perl* nécessaires pas toujours disponibles sur tous les sites

Problèmes rencontrés

- Changement du format de sortie de l'outil `oarnodes` à trois reprises
- Parallélisation complète du script Ivcmp impossible
- Bug du routeur *RENATER*
- Modules *Perl* nécessaires pas toujours disponibles sur tous les sites

Problèmes rencontrés

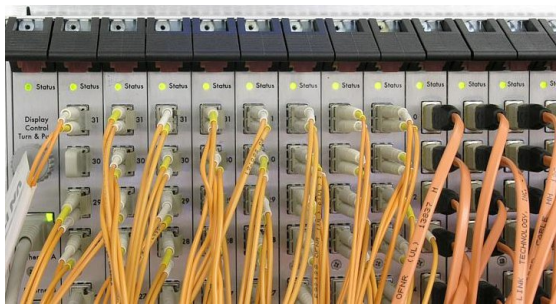
- Changement du format de sortie de l'outil `oarnodes` à trois reprises
- Parallélisation complète du script Ivcmp impossible
- Bug du routeur *RENATER*
- Modules *Perl* nécessaires pas toujours disponibles sur tous les sites

Partie III

Les technologies liées aux clusters

Infiniband

- **Protocole libre**, supporté par défaut par les noyaux Linux récents
- De **10Gbps** à **20Gbps**
- Propose un **mode TCP/IP** sur Infiniband, pour la compatibilité
- Peut être relié à des **câbles cuivrés** ou des brins de fibre optique
- **De plus en plus utilisé** par les supercalculateurs



Myrinet

- **Protocole propriétaire** conçu par Myricom
- Technologie **destinée aux clusters**
- Jusqu'à **10Gbps de débit** depuis 2006
- Une **latence exemplairement faible**
- Un coût important (deux brins de **fibre optique** par carte, commutateurs)
- **De moins en moins de supercalculateurs** utilisent Myrinet

Message Passing Interface

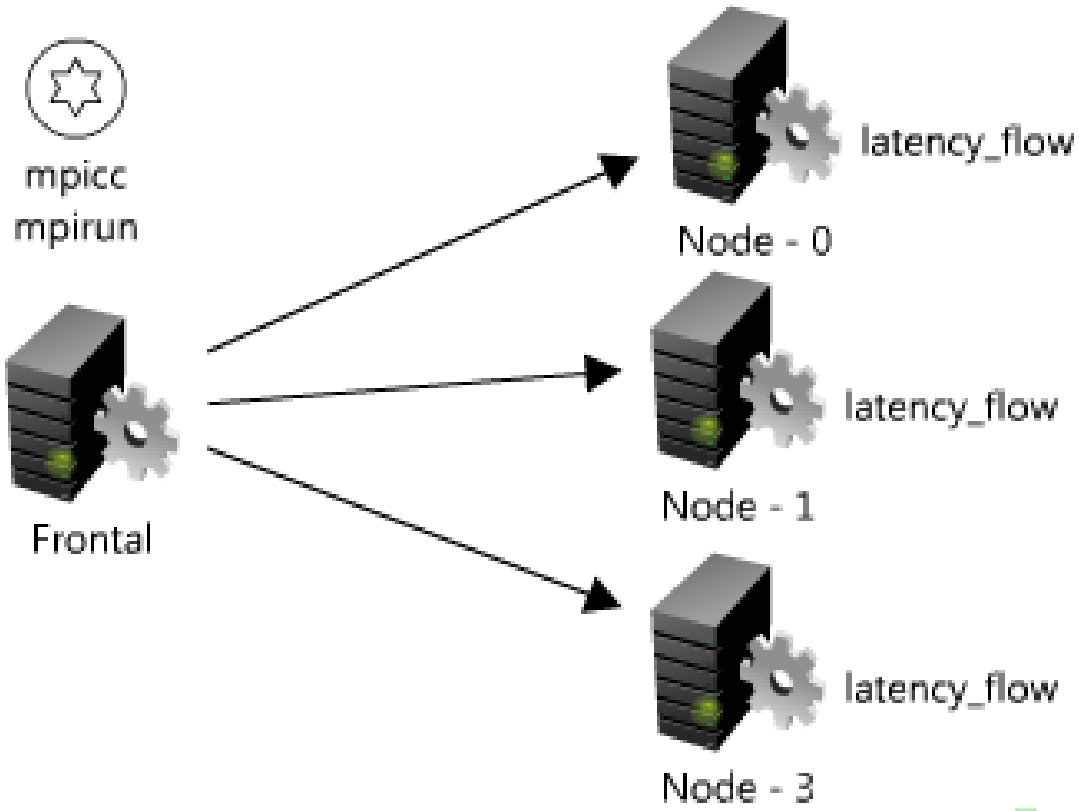
- Initialement prévue pour le *C/C++* et *Fortran*.
- Le Forum MPI, constante évolution.
- Portabilité, multi-plateforme.
- Prédestiné au monde du HPC.
- Des librairies et fonctions
  - Communicateurs
  - Fonctions point à point
  - Fonctions collectives
  - Types dérivés

MPI : Fonctionnement (1/3)

Lancement du même programme depuis le frontal.

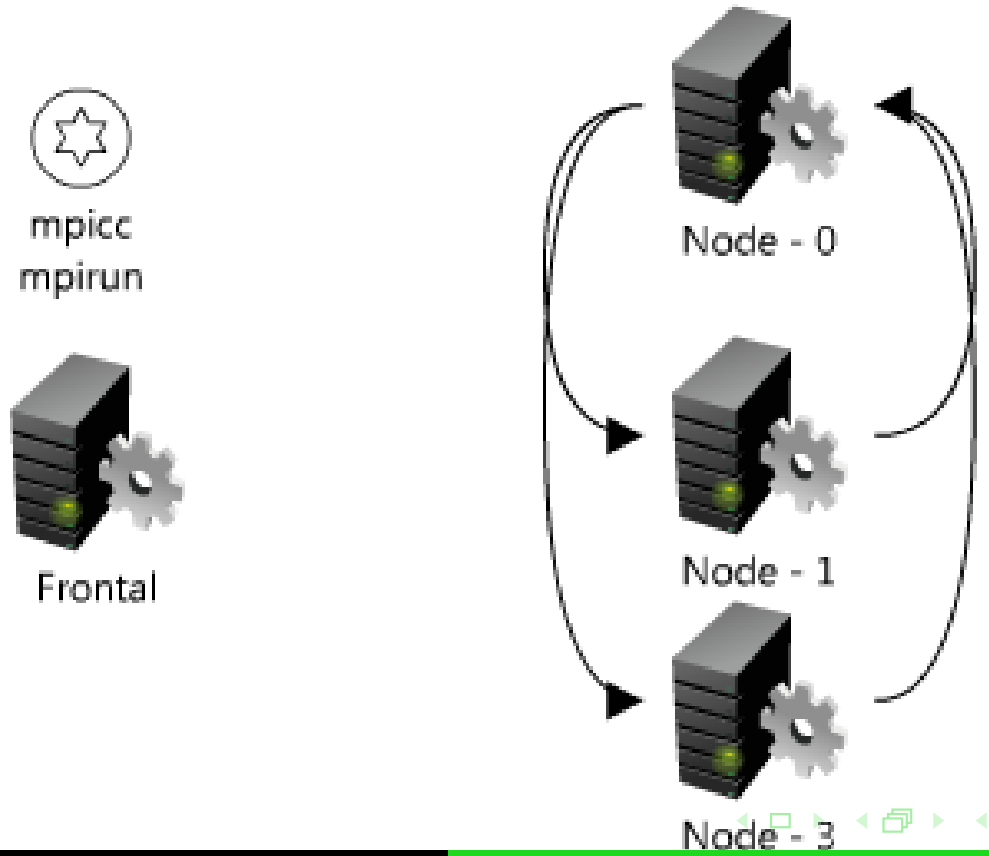
MPI : Fonctionnement (2/3)

Lancement en parallèle sur chaque nœud.



MPI : Fonctionnement (3/3)

En fonction du rang, orientation différente.



MPI : Historique

- Juin 1994 : la **version 1.0** de MPI est lancée.
- Juin 1995 : apparition de la **version 1.1**.
- 1997 : mise en conformité et **version 1.2**.
- Juillet 1997 : **version 2.0** ajouts de fonctionnalitées.
- Septembre 2008 : clarifications et **version 1.3**.
- Juin 2008 : clarification en vue de la **version 2.1**.
- 2009 : corrections, **version 2.2**.
- Futur 2010 : **version 3.0**, et nouvelles fonctionnalités

Implémentations

Il existe trois principales implémentations de MPI2.

- LAM
  - Ancienne implémentation
  - Laisse place à OpenMPI
- OpenMPI
  - Libre et Open-Source
  - Support d'Infiniband et Myrinet (re-compilation)
- MPICH2
  - Libre et Open-Source
  - Support d'Infiniband
  - Paquet spécial Myrinet (MPICH2-MX)

Utilisation

- Compilation
  - Spécifique selon l'implémentation.
  - *mpicc.openmpi mpicc.mpich*
- Exécution
  - Spécifique selon l'implémentation et donc du site.
  - *mpirun.openmpi mpirun.mpich*
- Comportement
  - Différent, options non normalisées.
  - *–mca btl*

Partie IV

Les tests de débits/latences

Les Solutions testées (1/3)

- OSU Micro-Benchmarks (OMB)
- Des tests multiples basés sur MPI

```
# OSU MPI Bandwidth Test v3.1.1
# Size          Bandwidth (MB/s)
1                         2.00
2                         4.06
4                         8.16
8                        16.50
16                       32.89
```

Les Solutions testées (2/3)

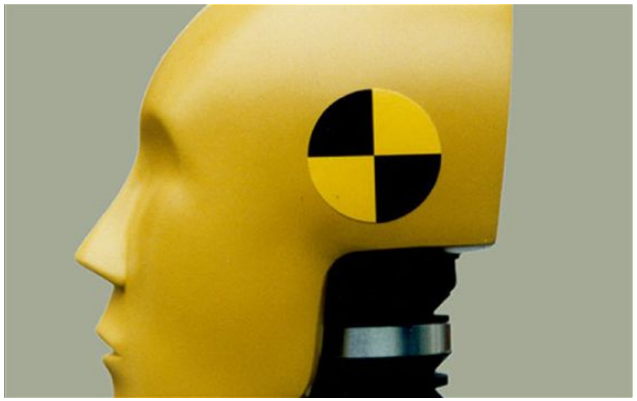
NetPIPE

- Network Protocol Independent Performance Evaluator
- Des tests utilisant MPI

```
0: gdx-102
1: gdx-53
Now starting the main loop
  0:       1 bytes  21388 times -->      1.85 Mbps in      4.12 usec
  1:       2 bytes  24249 times -->      3.72 Mbps in      4.10 usec
  2:       3 bytes  24408 times -->      5.56 Mbps in      4.12 usec
  3:       4 bytes  16199 times -->      7.46 Mbps in      4.09 usec
```

Les Solutions testées (3/3)

- Les Tests alternatifs
  - Infiniband
  - Myrinet



Notre programme : Objectifs (1/3)

1. **Mesurer le débit et la latence** entre chaque nœud d'un cluster
2. Donner la possibilité de faire des **bissections** et d'en mesurer le débit total constaté
3. Dresser une **matrice** de ces résultats et proposer une sortie **YAML**
4. Proposer un script de conversion du YAML en **tableau HTML coloré**
5. Proposer un **script d'automatisation** des tests de bissections consécutives
6. Proposer la possibilité d'en faire un **graphique**

Notre programme : Objectifs (2/3)

- Un unique programme d'environ **mille lignes de C**, utilisant la technologie **MPI**
- Un script **Bash** pour l'automatisation
- Un script **Ruby** pour la conversion en HTML coloré
- Un script `gnuplot` pour le graphique

Notre programme : Objectifs (3/3)

```
[redacted]@grelon-77:~/tests_debit_latence$ sort -u $OAR_NODEFILE > machines && mpirun.openmpi -np $(cat machines | w

OAR FLOW LATENCY TESTS
----------------------
       -s <n>K   : Message size for flow tests exchanges, in bytes (with K, M, or G suffix). Min 64K, default 1M.
       -p <n>    : Tests precision (repeats each test <n> times and make averages). Default 10.
       -b        : Bisection (first node of the first half with the first node of the seconde half, and so on).
       -rb, -r   : Bisection, with random pairs.
       -bg, -g   : Bisection, with gnuplot coordinates output.
       -o <file> : YAML output.
       -h        : This help.

AUTHORS : <julien@vaubourg.com>
          <sebastien.badia@gmail.com>

[redacted]@grelon-77:~/tests_debit_latence$
```

Le mode matrice (1/3)

Latence Temps mis par un paquet vide pour transiter (μs)

Débit Temps mis par un octet pour transiter (Mo/s)

1. Un **maître**
2. Les autres **écoutent**
3. Choix d'un **envoyeur**, distribution des **rôles**
4. Échanges des **tests**
5. **Répétition** des tests
6. **Changement** d'envoyeur, etc.
7. Sortie de la **matrice**

Le mode matrice (2/3)

	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
```

Le mode matrice (3/3)

```
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
```

La sexy matrice

- Mise en valeur des extrémités
- Mise en valeur des quartiles remarquables

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

Le mode bissection (1/3)

- Mesurer le **débit total maximum** possible sur le réseau
- **Détecter les nœuds** qui ne sont pas connectés
- Paires formées **aléatoirement**

Le mode bissection (2/3)

1. Un **maître**, les autres **écoutent**
2. Distribution de tous les **rôles**
3. Préparation et mise en **attente**
4. **Coup de feu** : échanges des tests
5. **Répétition** des tests (ensemble)
6. Envoi de tous les **résultats**, sortie de la matrice

Le mode bissection (3/3)

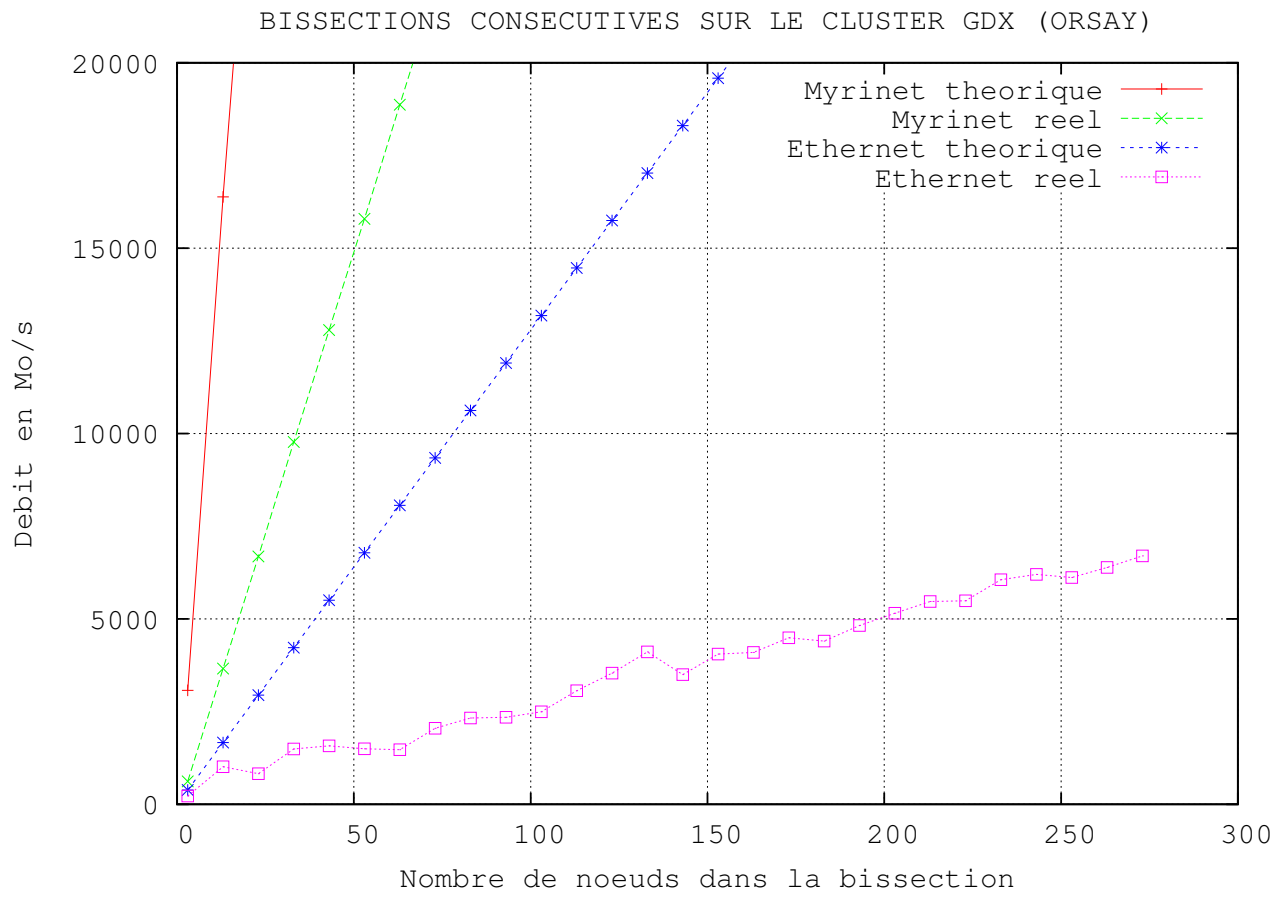
```
+--------------------+--------------------+
| From gdx-212       | To gdx-214         |
|                    |          17,118 us |
|                    |       633,103 Mo/s |
+--------------------+--------------------+
| From gdx-213       | To gdx-215         |
|                    |          17,011 us |
|                    |       631,490 Mo/s |
+--------------------+--------------------+

Latency :
Max : 17,118 us        From gdx-212 to gdx-214
Min : 17,011 us        From gdx-213 to gdx-215
Sum : 34,130 us
Avg : 17,065 us

Flow :
Max : 633,103 Mo/s     From gdx-212 to gdx-214
Min : 631,490 Mo/s     From gdx-213 to gdx-215
Sum : 1264,593 Mo/s
Avg : 632,297 Mo/s
```

(YAML possible)

Les bissections consécutives (1/2)

- Mesurer la **charge du réseau** en fonction du nombre de nœuds qui communiquent
- **Comparer** les performances entre les types de réseaux
- Comparer les **débits théoriques** avec les débits réels
- Détecter les goulots d'**étranglement**

Les bissections consécutives (2/2)



Problèmes rencontrés

Implémentations de MPI différentes selon les clusters

Impossibilité de faire **communiquer** les nœuds en MPI par *SSH*

Erreurs dûes à la compilation des implémentations

Absence de MPI sur certains clusters

Pas de support des **réseaux rapides** partout

Problèmes rencontrés

Implémentations de MPI différentes selon les clusters

Impossibilité de faire **communiquer** les nœuds en MPI par *SSH*

Erreurs dûes à la compilation des implémentations

Absence de MPI sur certains clusters

Pas de support des **réseaux rapides** partout

Problèmes rencontrés

Implémentations de MPI différentes selon les clusters

Impossibilité de faire **communiquer** les nœuds en MPI par *SSH*

Erreurs dûes à la compilation des implémentations

Absence de MPI sur certains clusters

Pas de support des **réseaux rapides** partout

Problèmes rencontrés

Implémentations de MPI différentes selon les clusters

Impossibilité de faire **communiquer** les nœuds en MPI par *SSH*

Erreurs dûes à la compilation des implémentations

Absence de MPI sur certains clusters

Pas de support des **réseaux rapides** partout

Problèmes rencontrés

Implémentations de MPI différentes selon les clusters

Impossibilité de faire **communiquer** les nœuds en MPI par *SSH*

Erreurs dûes à la compilation des implémentations

Absence de MPI sur certains clusters

Pas de support des **réseaux rapides** partout

Partie V

Un environnement rapide

Infiniband

- Chargement des **modules**
- Mise à jour vers un **noyau récent**
- Installation de **OpenMpi**
- Création d'un **tutoriel PDF** complet (installations, mise à jour des paquets)

Myrinet (1/2)

- **Patch** du noyau
- Indisponible dans les **paquets Debian**
- Pas de **service** pour lancer la création de l'interface
- Pilotes **propriétaires** (à demander)

Myrinet (2/2)

- Recupération des **pilotes**
- Recompilation d'un **noyau récent**
- Construction d'un **paquet Debian du noyau**
- Construction d'un **paquet des drivers Myrinet**
- Construction d'un **paquet OpenMPI Myrinet**
- Création d'un **service** de gestion du support de Myrinet
- Création d'un **tutoriel PDF** complet (installations, mise à jour des paquets)

Problèmes rencontrés

- Impossible de récupérer **MPICH2-MX**
- Installation de **drivers propriétaires** dans l'environnement *(sic)*
- Environnement **impossible à déployer** sur certains clusters
- **Déploiements hasardeux**, de façon générale, avec `kadeploy3`
- **Mise à jour du noyau problématique** (trois déploiements)

Partie VI

Problèmes généraux

Problèmes généraux

Sortie en *SSH* impossible depuis l'IUT

Documentation Grid5000 pas toujours à jour (documentation de l'outil `kaconsole3` pour Grenoble par exemple)

Maintenances fréquentes sur les *clusters* d'où des difficultés parfois à travailler correctement

Problèmes généraux

Sortie en *SSH* impossible depuis l'IUT

Documentation Grid5000 pas toujours à jour (documentation de l'outil `kaconsole3` pour Grenoble par exemple)

Maintenances fréquentes sur les *clusters* d'où des difficultés parfois à travailler correctement

Problèmes généraux

Sortie en *SSH* impossible depuis l'IUT

Documentation Grid5000 pas toujours à jour (documentation de l'outil `kaconsole3` pour Grenoble par exemple)

Maintenances fréquentes sur les *clusters* d'où des difficultés parfois à travailler correctement

Partie VII

Conclusion

Conclusion (1/4)

Objectifs remplis :

- **Inventaire** automatique du matériel
- **Tests des disques durs**
- **Tests des débits et latences** des nœuds d'un cluster (vérification des correctes interconnexions)
- **Bissections** sur les clusters
- Support des **réseaux rapides** dans un environnement personnalisé

Conclusion (2/4)

Pour cela, les administrateurs bénéficient à présent :

- De plusieurs **nouveaux outils** entièrement paramétrables
- De **sorties YAML**, de matrices non-parsables, de matrices colorées, de graphiques, de tutoriels PDF

Nous avons aussi apporté :

- De nombreux **problèmes de matériel détectés** et vérifiés
- De nombreux **bugs rapportés**

Conclusion (3/4)

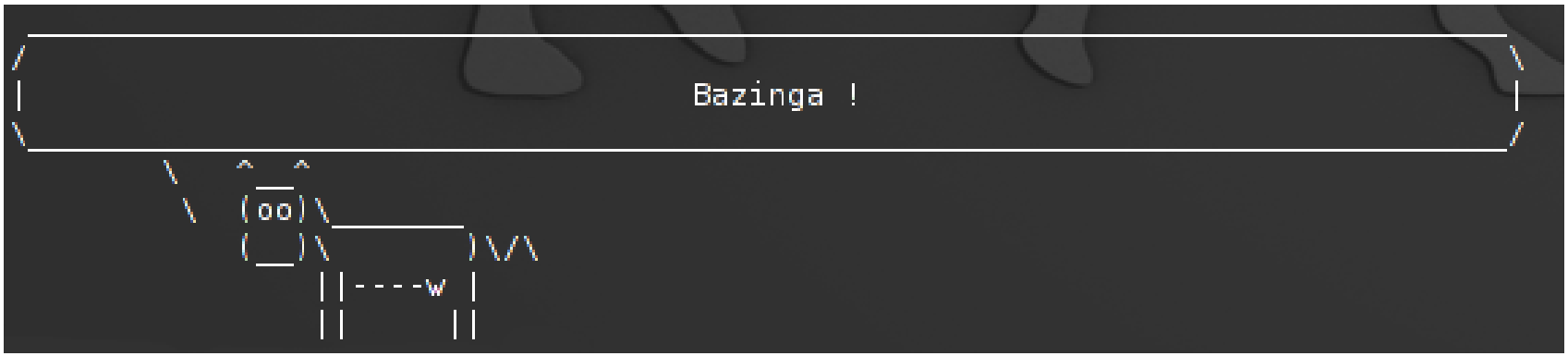
D'ores et déjà une **vidéoconférence** a eu lieu au LORIA (siège nancéen de Grid5000) pour un administrateur d'un autre site intéressé par le script d'inventaire.

Conclusion (4/4)

Bénéfices personnels

Découverte du monde des **grilles de calcul** et des **clusters**
Approfondissement des langages *C*, *Perl*, *Ruby*
Découverte de la programmation en MPI
Organisation pour un travail en équipe conséquent

Et comme c'est bientôt Pâques



(*Easter egg* inside)