SIDUS : dédupliquer pour servir la reproductibilité

Emmanuel Quémener

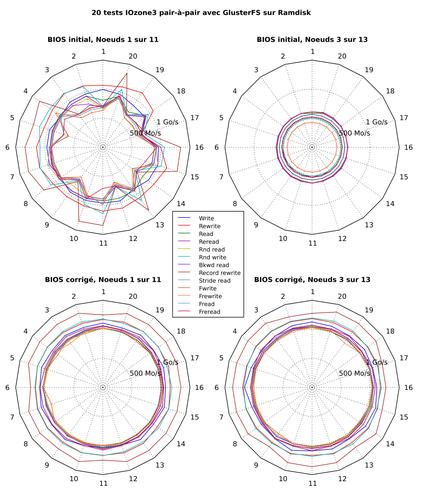
Centre Blaise Pascal, ENS-Lyon, Lyon, France

Reproductibilité : comment supprimer toute variabilité de la couche “système” ?

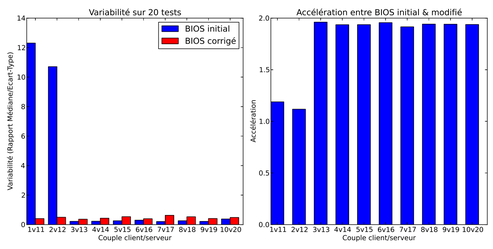
Étude : GlusterFS comme `scratch` en HPC

- **L'objectif** : évaluer les performances de GlusterFS
- **Le contexte** : Equip@Meso, l'heure des choix
  - Nœuds sans disque local
  - NFS robuste mais limité en performances
  - Solution propriétaire inabordable
  - Solution HPC peu intégrable à Debian
  - Solution CephFS trop avant-gardiste
- **Le banc d'essai :** 20 nœuds, 2 réseaux, 1 OS, 1 test
  - 20 HP SL230 bi-SandyBridge avec 64 Go RAM
  - Réseaux Gigabit Ethernet & InfiniBand FDR
  - Couche contrôleur I/O supprimée par usage Ramdisk
  - GlusterFS en *IP over IB*
  - **Socle logiciel SIDUS**
- **Les expériences :** 20 tests, 10 clients sur 10 serveurs
  - Utilisation comme Ramdisk de TMPFS ou de BRD
  - Test de système de fichiers IOzone3
  - 10 couples client/serveur
- **Les résultats :** faible performance ou forte variabilité
  - performance autour de 1 Go/s de transfert
  - forte variabilité, bonne performance sur 2 couples
  - faible variabilité, faible performance sur 8 couples
- **Une interrogation :** quelles sources de variabilité ?
  - Nœuds identiques, interconnexions identiques
  - **Même OS au bit près par SIDUS**
  - Seule différence possible : le BIOS
- **La conclusion :** le BIOS responsable !
  - réglages d'usine délétères en performance (C-States)
  - mixage de réglages catastrophique en reproductibilité

IOzone3 sur 2 client/serveur avant & après réglages de BIOS



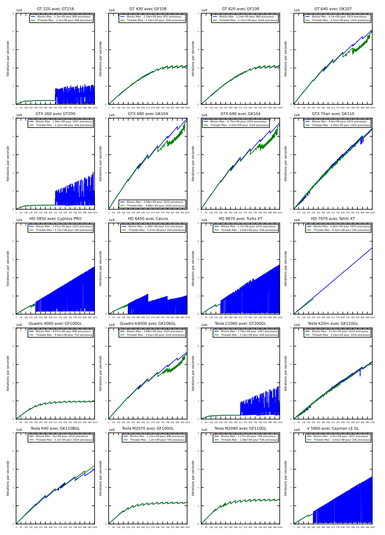
Variabilité & Performance pour les 10 client/serveur avant & après réglages de BIOS



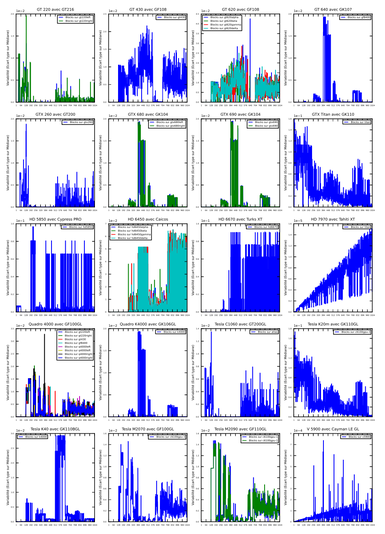
Étude : quel parallélisme massif des GPU ?

- **L'objectif** : évaluer le comportement en parallélisme des GPU
- **Le contexte** : contexte *many-cores*, comment comparer ?
  - Deux constructeurs “sérieux” : Nvidia & AMD
  - Deux langages : CUDA pour Nvidia, OpenCL pour tous
  - Deux parallélismes : *Blocks/WorkItems* & *Threads*
- **Le banc d'essai :** 26 stations/nœuds, 20 GPU différents, 1 OS
  - Dell C6100, Precision T7500, T5600, T7600, Optiplex 745
  - De 1 à 2 GPU différents dans chaque hôte
  - 20 GPU de référence différente : Nvidia, AMD
  - 3 catégories d'usage : bureautique, *gaming*, *HPC*
  - **Socle logiciel SIDUS**
- **Les expériences :** 1 OS, 1 langage, 1 API, 1 test
  - Langage OpenCL appelé en Python/Numpy
  - Programme test : Pi par Monte Carlo
  - Cœur de calcul comprenant un cycle de 32 opérations :
    - 24 en entier (essentiellement la MWC de Marsaglia)
    - 5 en virgule flottante simple précision
    - 2 de test
    - 1 de branchement
  - Parallélisation par distribution des itérations
- **Une interrogation :** quelles sources de variabilité ?
  - Nœuds ou stations identiques
  - **Même OS au bit près par SIDUS**
- **Les conclusions :** test simple mais instructif !
  - Performance : 1024 tâches faibles pour tester GPU récents
  - Comportement linéaire en parallélisme peu respecté
  - Comportement comparable pour des générations identiques
  - Variabilité : une signature aussi pertinente que la performance

Performance de 20 GPU



Variabilité de 20 GPU



SIDUS est “une instance unique distribuant un système d'exploitation universel”

S*ingle* **I***nstance* **D***istributing* **U***niversal* **S***ystem*

SIDUS : quelques mots clés pour un LiveCD reconfigurable partagé en réseau

- **PXE** : utilisation d'un démarrage en réseau
- **TFTP** : fourniture d'un noyau et d'un système de démarrage
- **AUFS** : superposition de systèmes en lecture seule et lecture/écriture
- **NFSROOT** : système racine unique partagé par tous les clients

SIDUS a deux principales propriétés

- **Unicité du système** : **tous** les clients démarrent exactement le **même** système (au bit près)
- **Usage des ressources locales** : les **processeurs** et **mémoire vive** exploités sont ceux des **clients**

SIDUS en 7 questions-réponses : `http://www.cbp.ens-lyon.fr/sidus/`

Pourquoi ?

- **Uniformiser** *de facto* tous les postes
- **Limiter l'administration** à UN système
- Assurer la **reproductibilité**
- **Rationaliser** l'usage des postes de travail

Pour Quoi ?

- **Nœuds de cluster** de calcul scientifique
- **Postes** de salle **libre-service**
- **Stations de travail** graphiques
- **Paillasses** d'**expérimentation** numérique
- **COMOD** : *Compute On My Own Device*
  - démarrage sur le poste complet
  - démarrage dans une machine virtuelle

Où & Quand ?

- **Centre Blaise Pascal : salle**
  - 12 clients légers boostés en mars 2010
  - 22 stations avec GPU différents fin 2013
- **Centre Blaise Pascal : cluster**
  - 24 nœuds en mars 2010
  - 76 nœuds permanents fin 2013
- **Centre de calcul PSMN, ENS-Lyon**
  - 100 nœuds mi 2012 en qualification
  - **400** nœuds en 2014 dont Equip@Meso
- **Laboratoires, ENS-Lyon**
  - Laboratoire de Chimie : été 2012
  - Laboratoires RDP, IGFL : fin 2013
  - Laboratoires LBMC, UMPA : début 2014
- **École de physique des Houches**
  - éditions depuis 2011 : 60 utilisateurs

Comment ?

Socle AUFS

- AUFS pour *Another Union File System*
- Un système NFSroot en lecture seule
- Un système TMPFS en lecture/écriture
- AUFS comme glue entre les deux systèmes

Installation : 8 étapes et 3 décisives

1 Formation d'un système par Debootstrap
6 Création de la séquence de démarrage
7 Importation noyau & initrd sur serveur TFTP

Administration simplifiée

- Passage dans l'instance par chroot
- Application des commandes « standard »
- Montage des dossiers « système » au besoin

Pour Qui ?

- **Chercheur** en informatique scientifique
- **Ingénieur** en calcul scientifique
- **Gestionnaire** de salle informatique
- **Formateur** avec des outils informatiques
- **RSSI**

Combien ?

- 1 réseau idéalement 1 Gb/s (débit disque)
- 1 serveur (virtuel) avec pour 100 clients :
  - 2 CPU
  - 8 Go de RAM,
  - 50 Go d'espace par architecture complète
- 1 personne motivée pendant 1 journée