



JoSy 2020

SIDUS : outil de reproductibilité

Un “grand pas” pour la déduplication
Un unique “petit pas” pour la reproductibilité ?

Emmanuel Quémener

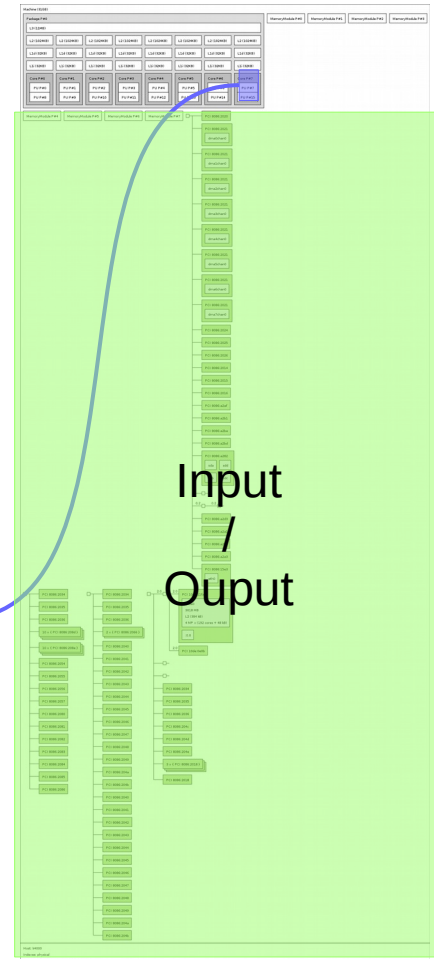
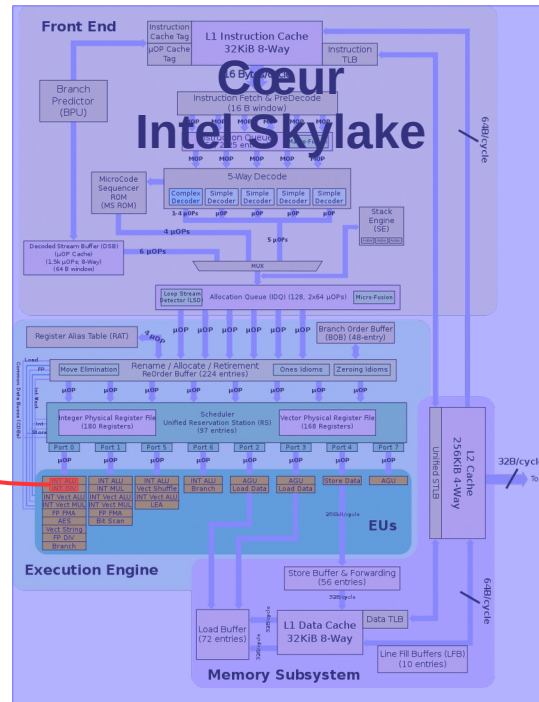
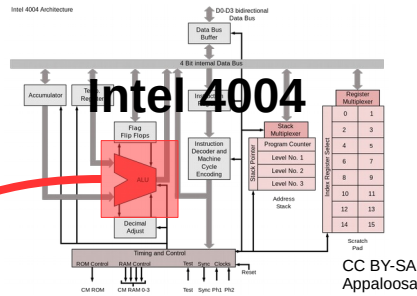
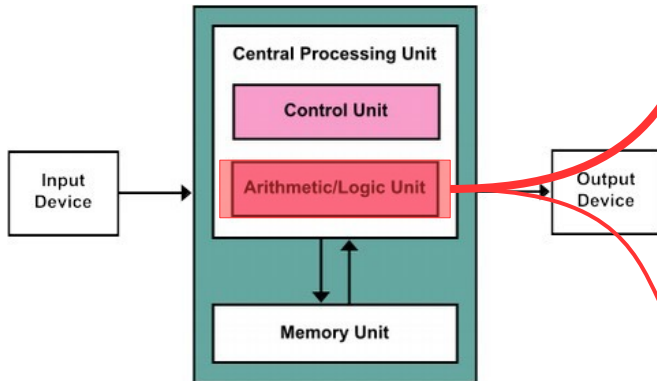
SIDUS : de la simplification... à la sophistication (extrême ?)

- Tout d'abord, une approche « pragmatique »
 - **Postes dans une salle de formation** : simplifier l'administration
 - **Nœuds d'un petit cluster** : s'assurer d'une reproductibilité (de conf')
- Ensuite, la montée en puissance :
 - **Dizaines de nœuds de cluster** : évaluer la scalabilité
 - **Du Centre d'essais au Mésocentre PSMN** : tester la portabilité
- Enfin, l'explosion des combinatoires matérielles :
 - **Différents CPU, GPU, architectures** : vérifier l'universalité de l'approche
 - **Tous les Plateaux techniques** : contrôler l'accessibilité à l'utilisateur.

La causalité & son corollaire...

- Énoncé : conditions, causes et effets...
 - « Dans les mêmes **conditions**,
les mêmes **causes** produisent les mêmes **effets** ».
- Conséquence du « déterminisme » : reproductibilité
 - **Mêmes** conditions, **mêmes** causes, **mêmes** effets...
- Problèmes :
 - Séparer les « conditions » et les « causes »
 - Déterminer les conditions initiales (le contexte)

Du « système ouvert » à Von Neumann : évolutions...



Reproductibilité :

qu'est-ce qu'un « même effet » ?

- Tout dépend du contexte :
 - « Système Informatique » : les « conditions »
- Tout dépend de l'effet :
 - Même valeur au bit près pour un résultat
 - Même résultat comparable (au sens « physique »)
 - **Même « travail » pour un résultat : temps*ressource**
- En HPC : intrication scalabilité & reproductibilité
 - Toutes les ressources sont parallèles : programmes parallélisés
 - Étude de la scalabilité : reproductibilité du temps d'exécution

L'informatique moderne : système de « compliqué » à « complexe »

- Compliqué : « cum plicare » (plier ensemble)
 - « le tout est la somme des parties » (Descartes)
- Complexe : « cum plexus » (tisser ensemble)
 - « impossible d'identifier tous les éléments, toutes les relations »
- Nos « systèmes » de calcul scientifique : le socle de nos « codes »
 - Dépendances logicielles : versions, compilateurs, bibliothèques, ...
 - Système d'exploitation : noyau, services, processus, supervision, ...
 - Nœud : processeur (cœur, L1/2/3, ALU, UC), mémoire, interfaces
 - Réseaux : haut débit/basse latence, bas débit/haute latence
- Assurer (un minimum la reproductibilité) pour rassurer (l'utilisateur) ?
 - Quelle marge de manœuvre pour l'opérateur ? Entre matériel & « usage » .
 - **Quelles solutions ? Offrir une « archive » du socle.**

Variabilité numérique espace/temps

Quelle marge de manœuvre ?

- **Temps** : même machine, instants différents ?
- **Espace** : même instant, machines différentes ?
- Les solutions :
 - Restauration d'une même image système :
 - Replicator, SystemImager, MondoRescue, ...
 - Kadeploy sur Grid'5000, GUIX, NIX, ...
 - Boot iSCSI avec Back Office Snapshot (sur LVM, ZFSonLinux, BtrFS)
 - Installation suivant le même protocole :
 - FAI, Kickstart, Debian-Installer Preseed, ...
 - **SIDUS** : *Single Instance Distributing Universal System*

Ce que SIDUS n'est pas... Mais ce qu'il partage avec eux !

SIDUS n'est pas !

- LTSP : Linux Terminal Server Project
 - Un serveur, une administration simplifiée des clients
- FAI, Kickstart, Debian Installer Preseed :
 - « Et la machine remplace l'opérateur pendant l'installation »
- LiveCD en réseau :
 - Une image ISO distribuée par le réseau

Mais SIDUS partage avec eux

- Boot PXE, (TFTP), NFSroot, (AUFS)

Les Deux Propriétés de SIDUS

Reproductibilité espace/temps

- Unicité de la configuration
 - Deux clients SIDUS : le même OS au bit près !
- Exploitation des ressources locales
 - Processeurs & mémoire vive exploités : ceux du client !
- Reproductibilité ? Pour une Instance SIDUS inchangée
 - Stabilité dans le temps (pour un même client)
 - Deux démarrages sur une même machine offrent le même système
 - Stabilité dans l'espace (pour deux clients différents)
 - Deux clients démarrant au même instant disposent du même système

SIDUS en 7 Questions : CQQCOQP Pourquoi ?

Pourquoi ?

- Uniformiser de facto tous ses « clients »
- Limiter l'administration à un unique système
- Comparer les matériels avec un socle unique
- Récupérer des fluides (Watts & BTU)
- Rationaliser l'usage des postes de travail
- Investiguer un système sous anesthésie
- **Assurer la reproductibilité sur l'OS & ses applications**

SIDUS en 7 Questions : CQQCOQP

Quoi ? Qui ?

Pour Quoi ?

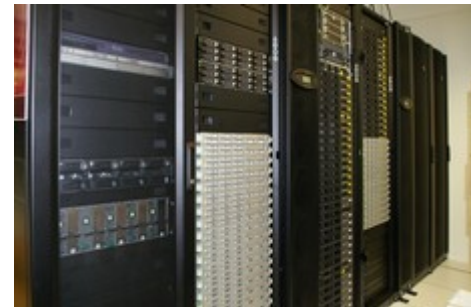
- Nœuds de cluster de calcul scientifique
- Postes de salle libre-service
- Stations de travail graphiques
- Paillasse d'expérimentation numérique
- COMOD : *Compute On My Own Device*

Pour Qui ?

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

SIDUS en 7 Questions : CQQCOQP, Où & Quand ?

- Centre Blaise Pascal, ENS-Lyon : salle
 - 12 Neoware en 2010Q1, 85 stations 2020Q4
- Centre Blaise Pascal, ENS-Lyon : cluster
 - 24 nœuds en 2010Q1, 176 nœuds 2020Q2
- Centre de calcul PSMN, ENS-Lyon
 - **100 nœuds 2012Q2, +600 nœuds 2020Q4**
 - Tout Equip@Meso
- Laboratoires, ENS-Lyon & ISA
 - Chimie, IGFL, Maths, Astrophysique
- École de physique des Houches
 - Éditions de 2011 à 2016



SIDUS en 7 questions : CQQCOQP

... la fin ! Comment ça marche ?

- AUFS : Another Union File System (ou OverlayFS)
 - Agréger des File Systems en un seul : astuce LiveCD
- 4 étapes :
 - Monter le NFSroot avec l'OS sur une racine
 - Monter un TMPFS sur une seconde racine
 - Utiliser la glue AUFS entre les deux
 - Offrir le résultat comme racine du système
- Comportement d'un FS en Read/Write normal
 - Au redémarrage, toute modification disparaît

Un prérequis : une « maîtrise » de chroot

SIDUS en 7 questions : CQQCOQP, la fin !

Comment ça s'installe : SIDUS en 8 étapes

- 1) Formation d'un système racine par Debootstrap
- 2) Création d'un « cordon ombilical » avec l'hôte
 - Montages des `/proc /sys /dev/shm`
- 3) Installation de dracut (& purge des paquets spécifiques)
- 4) Adaptation à l'environnement local
- 5) Pointage vers les services tiers
- 6) Création de la séquence de démarrage (AUFS ou OverlayFS)
- 7) Importation des noyau & initrd sur serveur TFTP ou HTTP
- 8) Détachement du système hôte

SIDUS en 7 questions : CQQCOQP, la fin !

Comment ça s'installe **sans** Debootstrap

- 1) Installation d'un OS fonctionnel sur un matériel
- 2) Rsync « intelligent » du système complet dans un dossier sur le NFS
 - Exclusion des dossiers `/sys`, `/proc`, `/dev`, `/tmp`, `/run`, `/var/run`, `/snap` et option « `--numeric-ids` »
- 3) Création d'un « cordon ombilical » avec l'hôte
- 4) Installation du « dracut » avec réseau et son paramétrage
- 5) Pointage vers les services tiers
- 6) Création de la séquence de démarrage (AUFS ou OverlayFS)
- 7) Importation des noyau & initrd sur serveur TFTP ou HTTP
- 8) Détachement du système hôte

SIDUS en 7 questions : CQQCOQP, ...la fin ! Comment ça s'administre ?

- Une limitation : un /proc doit être unique...
 - Forte vigilance sur les opérations qui « tapent dedans »
 - Manipulation de Java, compilation avec optimisation, installation de pilotes
- La Bonne :
 - Passage par chroot, opérations classiques directement
- La Brute :
 - Passage par chroot, mise en place du « cordon ombilical »
 - Opérations classiques, démontage du cordon
- La Truande :
 - Machine NFSroot en Read/Write, ...

SIDUS en 7 questions : CQQCOQP, ... la fin ! Combien ça coûte ?

- Un réseau « idéal » : Gigabit Ethernet (HD local)
 - Mais ça fonctionne en 100 Mb/s !
- Un serveur « idéal » : 4 cpu, 16 Go RAM, 10G, SSD
 - Mais ça fonctionnait avec un v(eau)40z pour 330 nœuds !
- Un client « idéal » : tous identiques
 - Mais ça fonctionne pour TOUTES les gammes au PSMN
- Un installateur/administrateur « idéal » : ;-)
 - Déployé par L. Taulelle à partir de rushs : PSMN
 - Déployé par T. Bellebois et C. Petit via doc en ligne : IGFL & PSMN

Usage de SIDUS en reproductibilité

Pourquoi certaines variabilités ?

- Pertinence de GlusterFS comme scratch distribué en HPC
 - Influence du BIOS dans les performances et la variabilité
- Influence de la localité sur les exécutions
 - « HyperThreading or not HyperThreading ? »
- Influence des conditions climatiques
 - De la nécessité de bien ventiler ses machines
- Les versions de pilote & conditions climatiques
 - Des variabilités « impulsionnelles » pour le meilleur ou pour le pire...

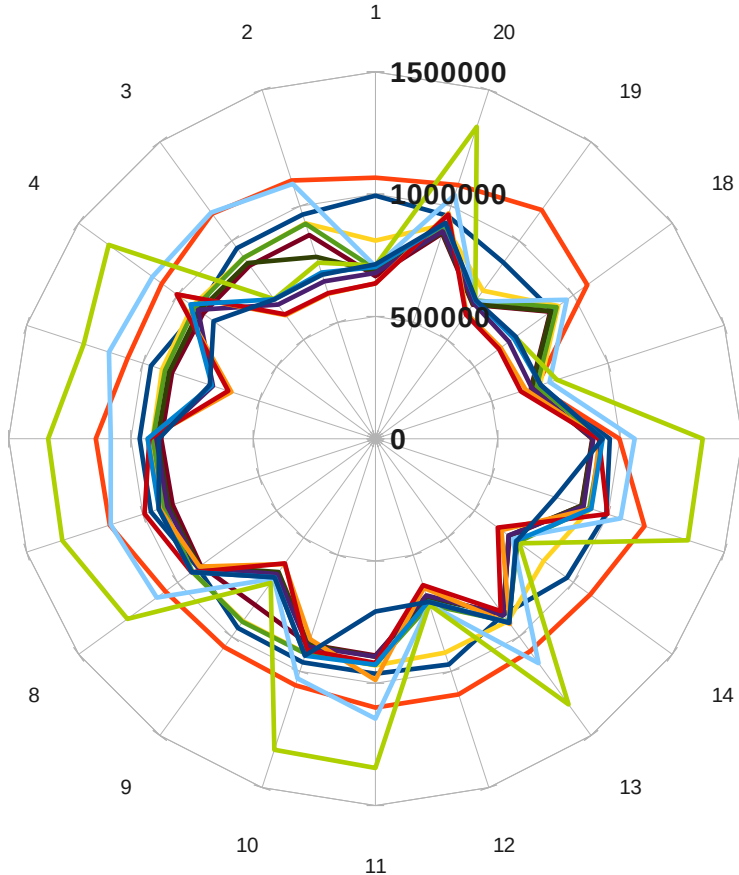
Manque de reproductibilité ?

Illustration par l'exemple : GlusterFS/IOZone

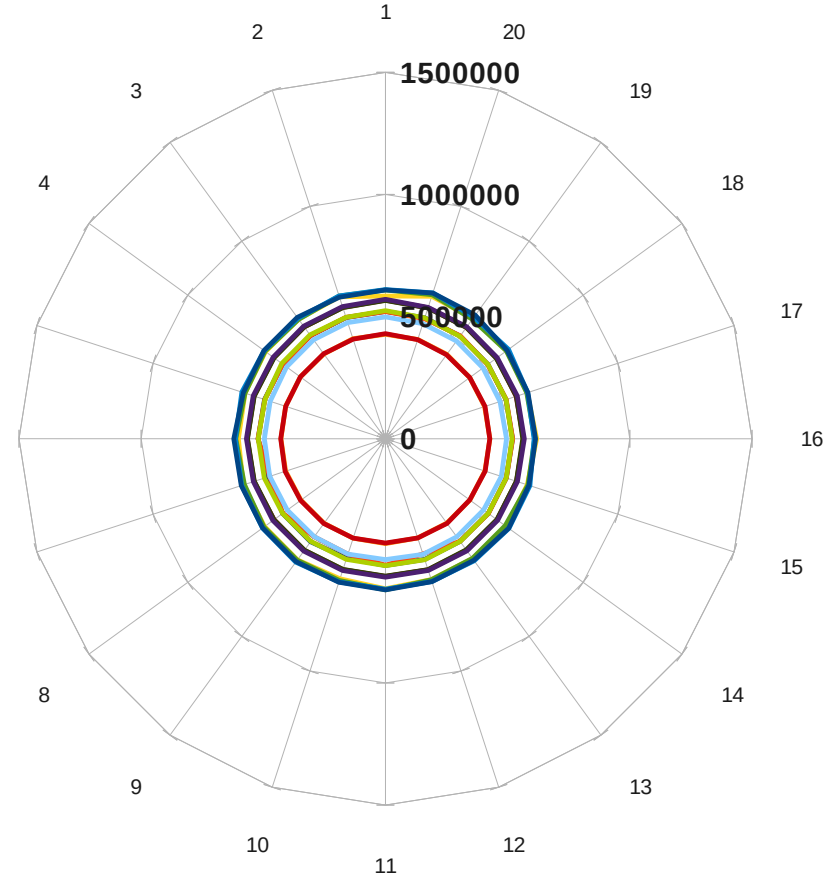
- Objectif :
 - Évaluation de GlusterFS comme /scratch de haute performance
- Plate-forme d'expérimentation : 20 nœuds + infrastructure
 - 20 nœuds Sandy Bridge 2x8 cœurs avec 64 GB de RAM
 - **Un système SIDUS Debian Wheezy**
 - Interconnexion InfiniBand FDR 56 Gb/s
 - Pas de latence disque : RamDisk BRD/Ext2 et TMPFS de 60 GB
 - 10 paires GlusterFS : 1 serveur sur RamDisk, 1 client
 - Usage de IOZone3 : 13 tests de lecture/écriture
 - 20 expériences pour un échantillon statistique représentatif

Jour 1 : lancement des tests & surprises ! Sur les vitesses d'exécution I/O

Nœud 11 vers Nœud 1



Nœud 13 vers Nœud 3

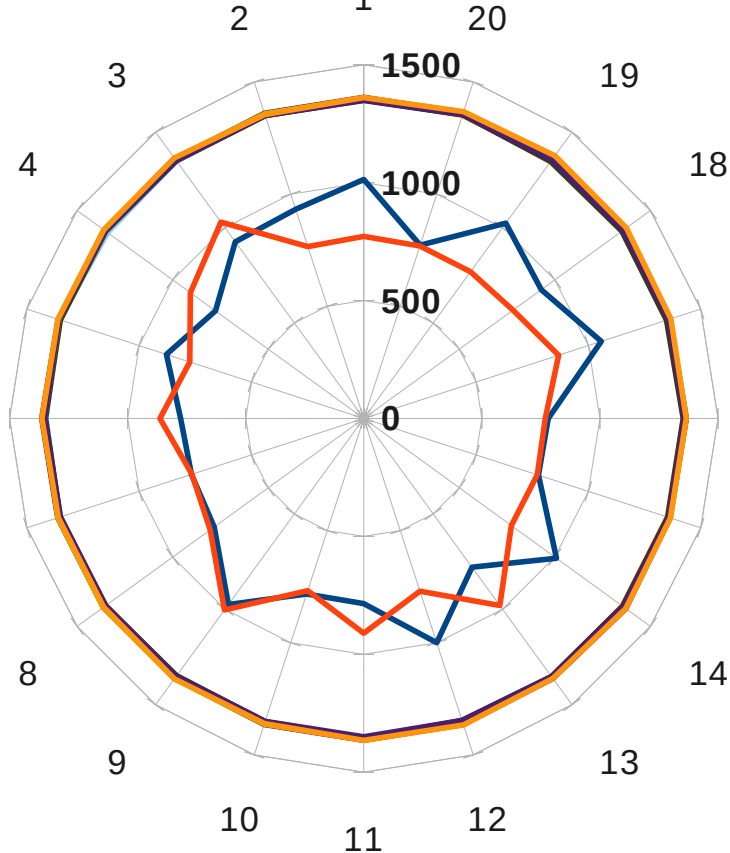


- Write
- Rewrite
- Read
- Reread
- Rnd read
- Rnd write
- Bkwd read
- Record rewrite
- Stride read
- Fwrite
- Frewrite
- Fread
- Freread

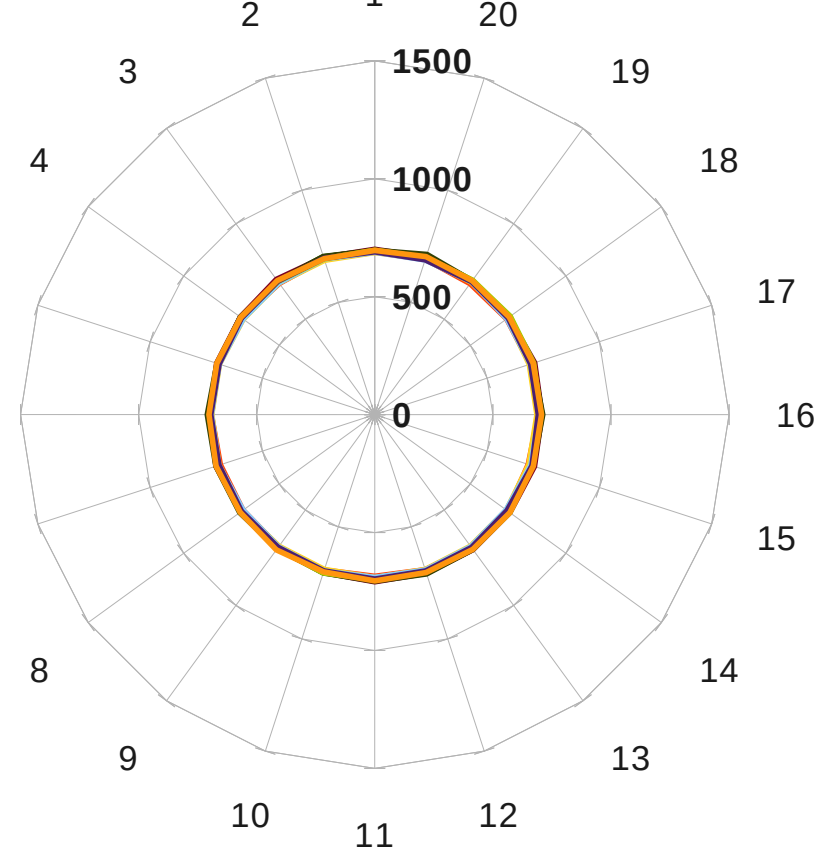
More is Better !

Jours 1 & 2 : modification & nouveaux tests Sur les durées d'exécution (*User Time*)

Pour les 10 couples, **avant...**



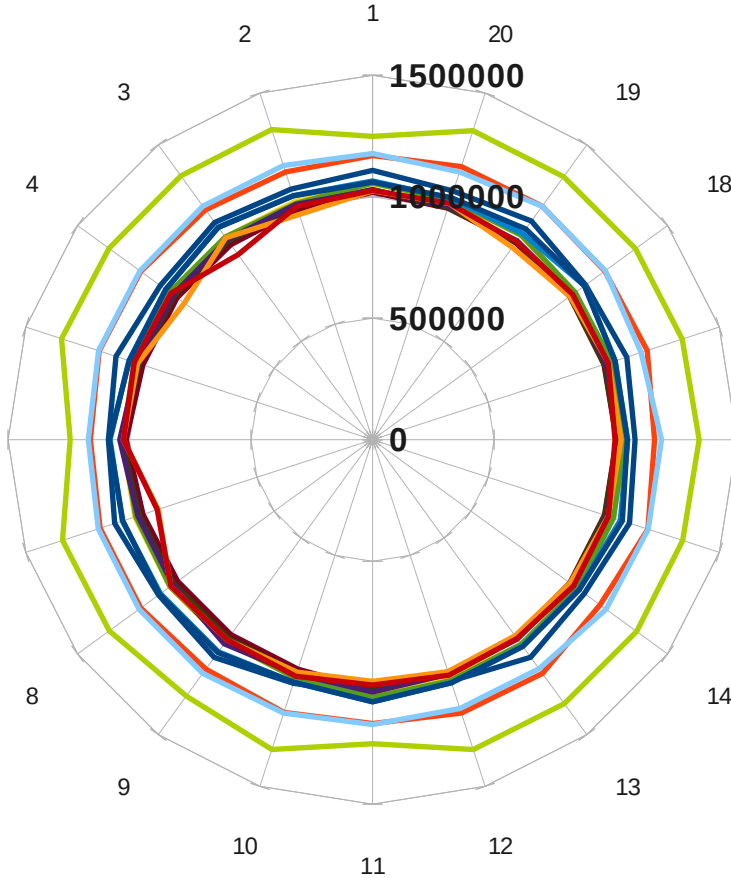
Pour les 10 couples, **après !**



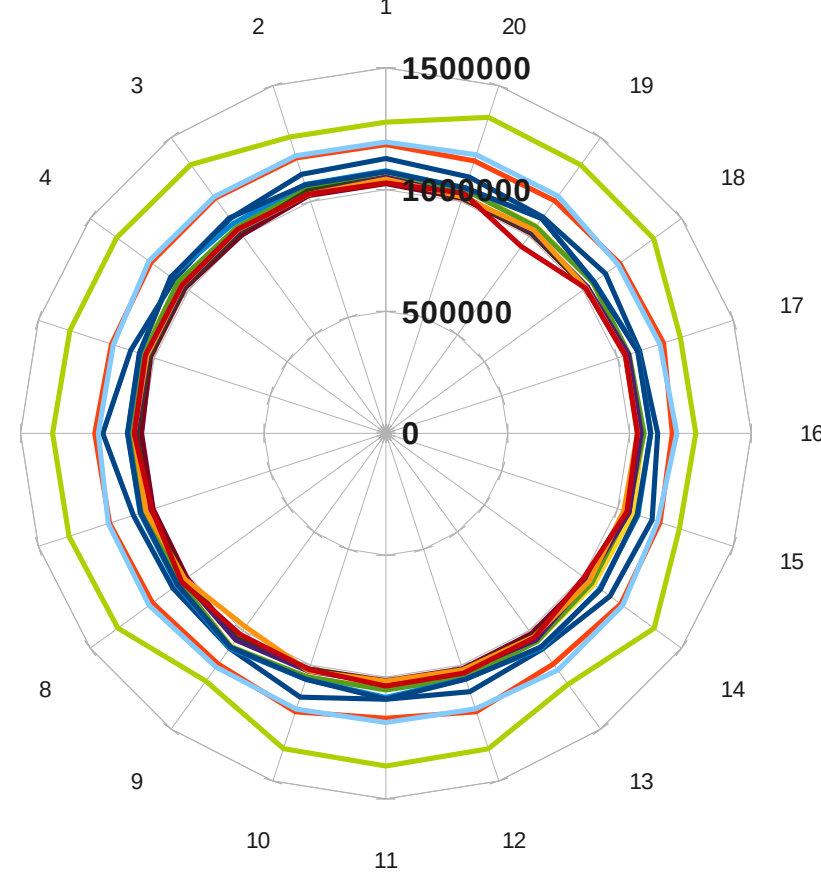
Less is Better !

Jour 2 : modification & nouveaux tests Sur les vitesses d'exécution I/O

Nœud 11 vers Nœud 1



Nœud 13 vers Nœud 3

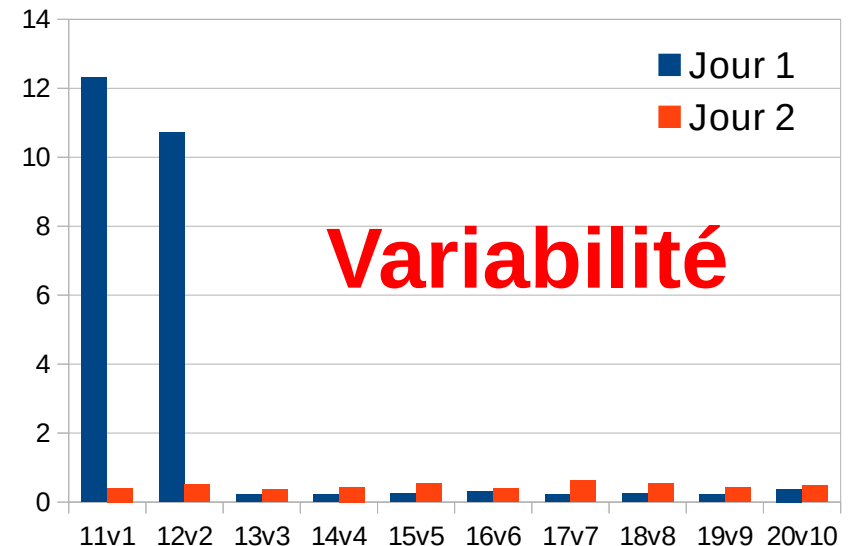
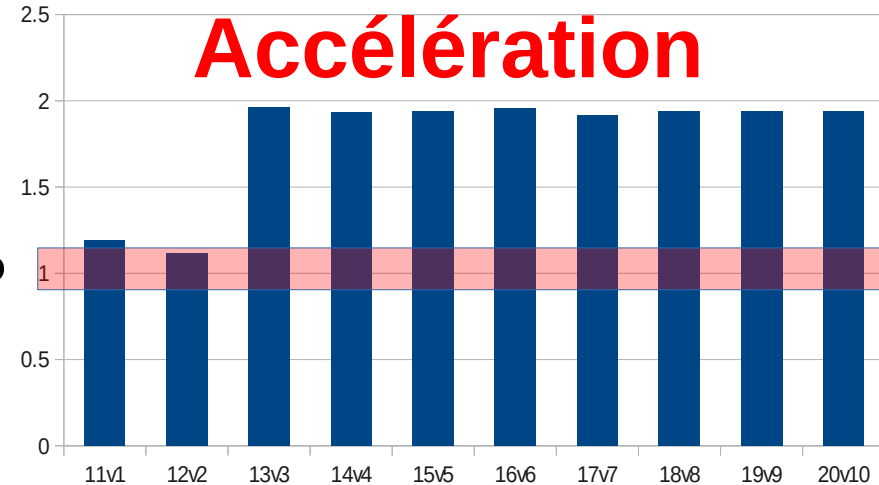


- Write
- Rewrite
- Read
- Reread
- Rnd read
- Rnd write
- Bkwd read
- Record rewrite
- Stride read
- Fwrite
- Frewrite
- Fread
- Freread

More is Better !

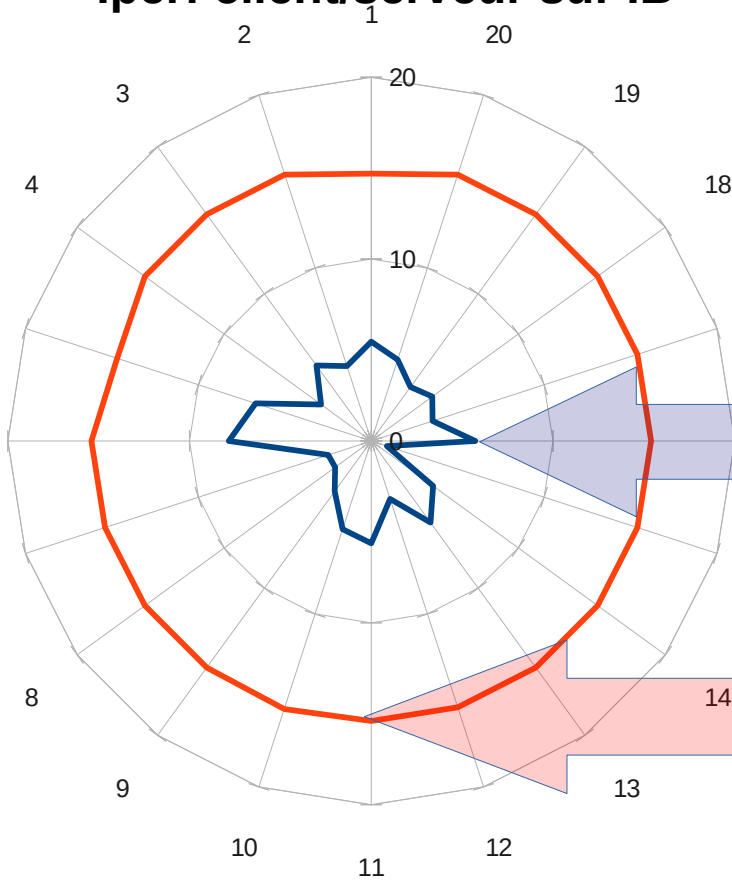
Quel miracle entre jours 1 & 2 ?

- Deux questions : Comment ...
 - ... multiplier par 2 la vitesse ?
 - ... diviser entre 20/30 sa variabilité ?
- La réponse :
 - Optimiser le réseau ? Non
 - Optimiser les noyaux des OS ? Non
 - Changer le BIOS ? OUI !!!
 - BIOS de 1 & 2 en Max Performance
 - BIOS de 3 à 20 par défaut
 - Solution : BIOS en Max Perf !

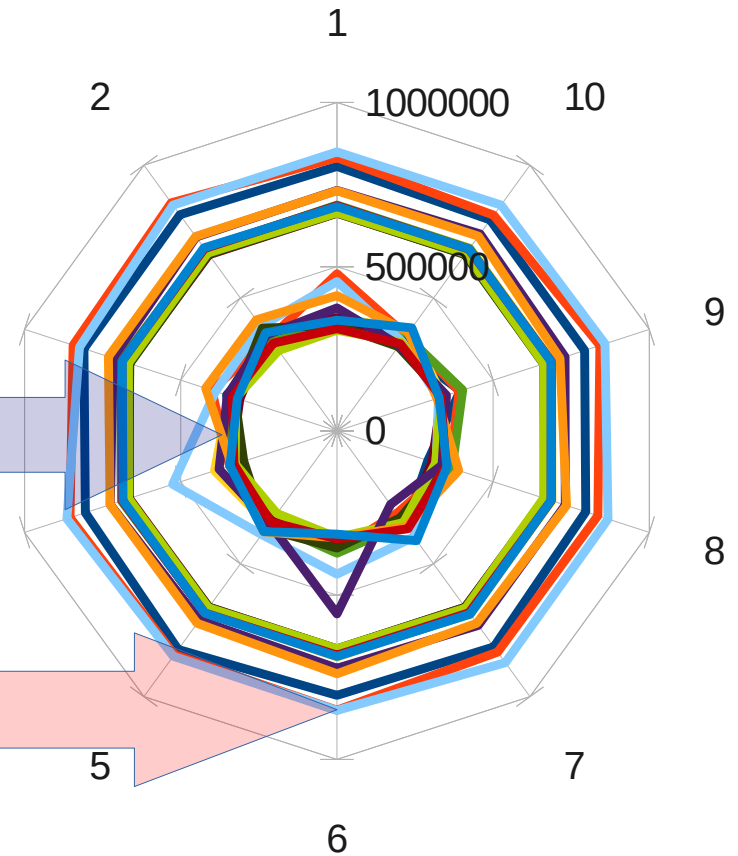


La non-reproductibilité reproductible ? Sur Equip@Meso

Iperf client/serveur sur IB



iozone3 sur GlusterFS



Avant

Après

Comportement de nœuds de cluster

Quelle variabilité en *Embarrassing Parallelism*

- Objectif :
 - Évaluer le passage à l'échelle en MPI, quelle statistique prendre ?
- Banc d'essai :
 - 48 nœuds bi-sockets quadri-cœurs R410, interconnexion Infiniband
 - **Systeme SIDUS unique**
 - Code Pi Monte Carlo distribution en MPI (10^{14} itérations)
 - Lancement de mpirun -np 384
 - 1000 simulations

Un client qui se parallélise bien. Pi Monte Carlo

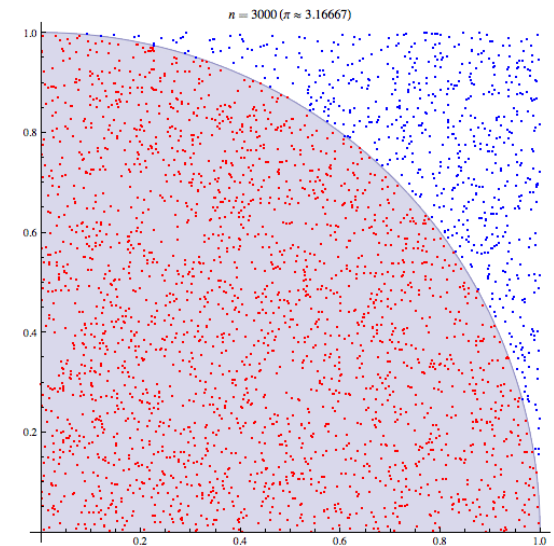
- Illustration classique de la méthode de Monte Carlo
- Implémentation : distribution des itérations

- De 2 à 4 paramètres

- Nombre total d'itérations
- Régime de Parallélisme : Parallel Rate (PR)
- (Type de variable : INT32, INT64, FP32, FP64)
- (RNG : MWC, CONG, SHR3, KISS)

- 2 observables simples :

- Estimation de Pi (juste indicative, Pi n'est pas rationnel :-)
- Temps écoulé





Certains le jugent non pertinent...

Coup d'œil sur un tutoriel du LLNL...

Introduction to GPU Parallel Programming

Data Heroes Summer HPC Workshop
June 27, 2016

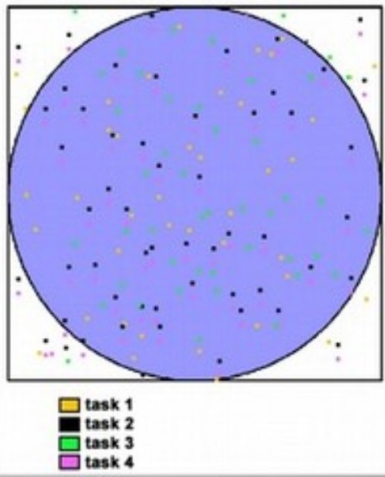
Donald Frederick,
Livermore Computing



LLNL-PRES-XXXXXX
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Approximation of Pi by Monte Carlo – Parallel Version

- Another problem that's easy to parallelize: All point calculations are independent; no data dependencies
- Work can be evenly divided; no load balance concerns
- No need for communication or synchronization between tasks
- Parallel strategy: Divide the loop into equal portions that can be executed by the pool of tasks
- Each task independently performs its work
- A SPMD model is used
- One task acts as the master to collect results and compute the value of Pi

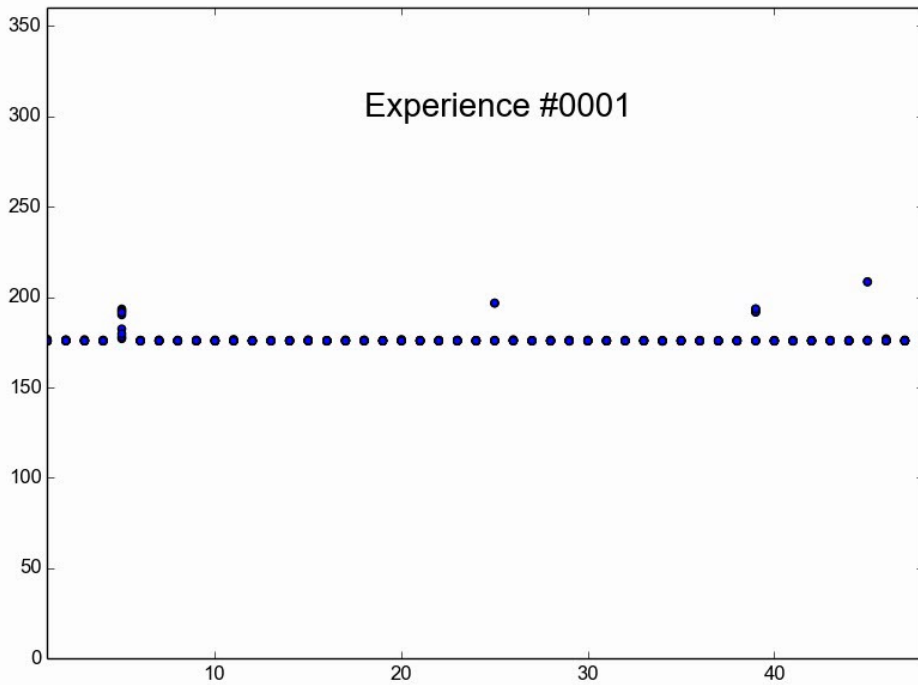


task 1
task 2
task 3
task 4

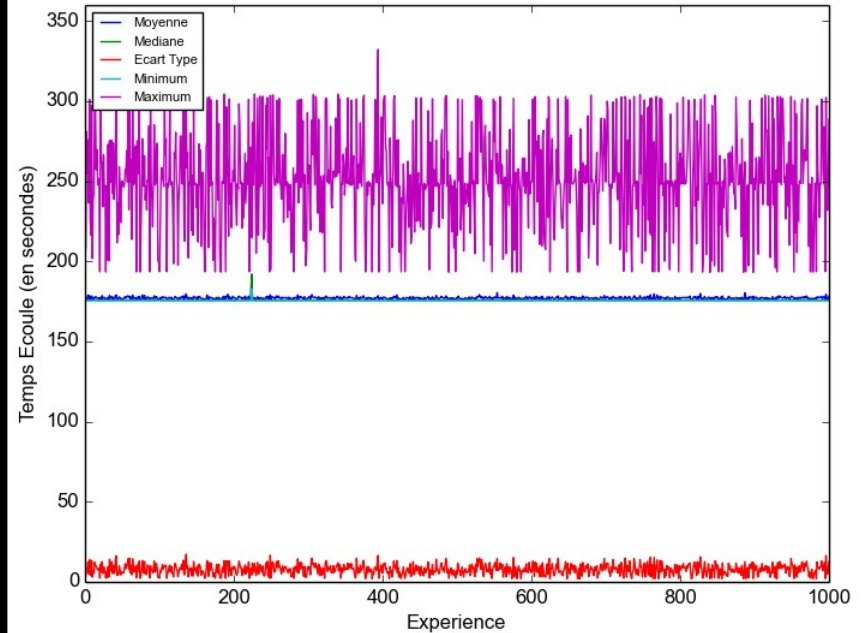
Lawrence Livermore National Laboratory

Distribution MPI sur 48 nœuds (384 cœurs) 1000 Tests et toujours des retardataires !

Temps écoulé en secondes



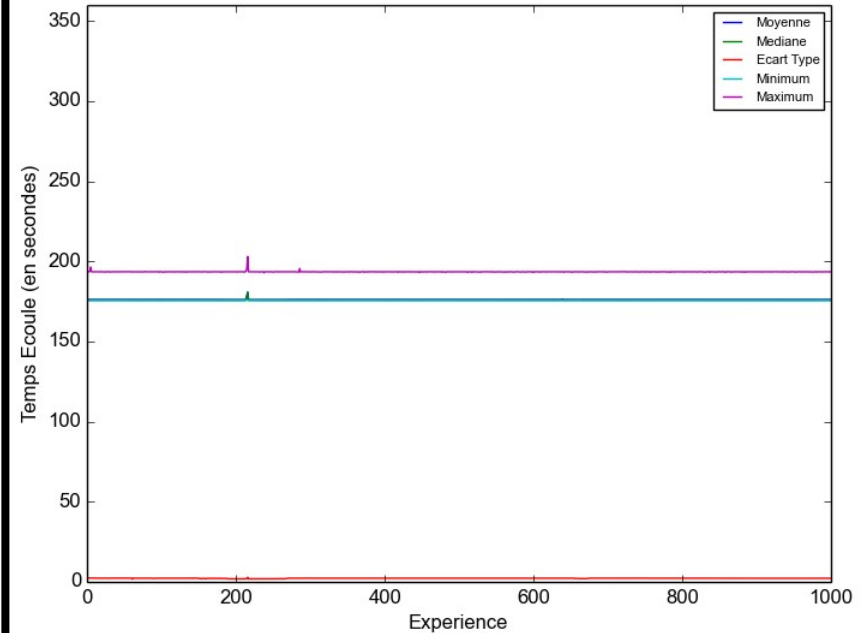
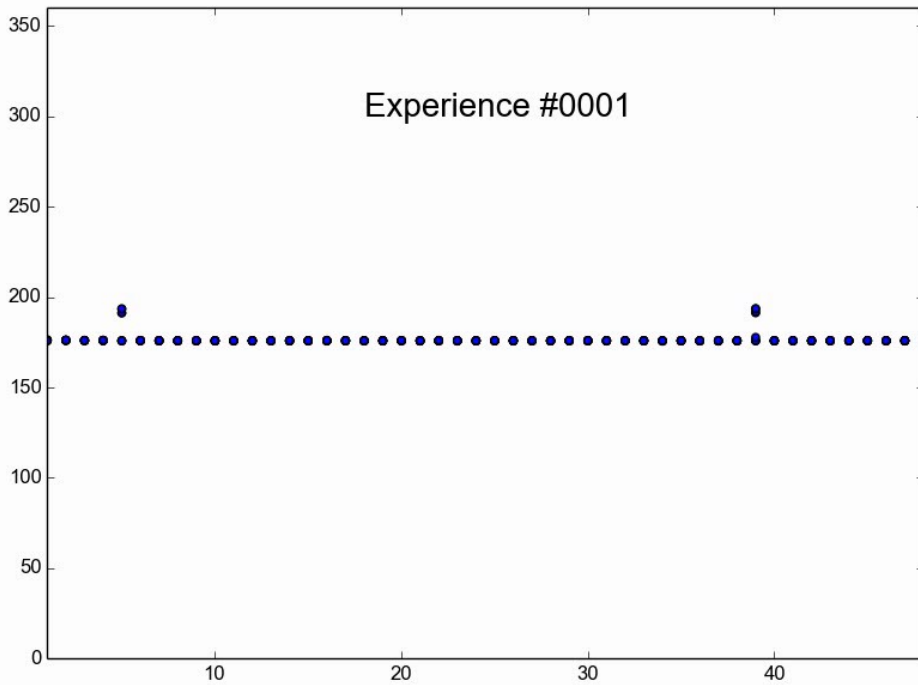
Indice du nœud utilisé (1 à 48)



Indice de chaque expérience

Distribution MPI sur 48 nœuds (384 cœurs) 1000 Tests et ça va beaucoup mieux !

Temps écoulé en secondes



Indice du nœud utilisé (1 à 48)

Indice de chaque expérience

Pourquoi cette variabilité sur le jour 1 ?

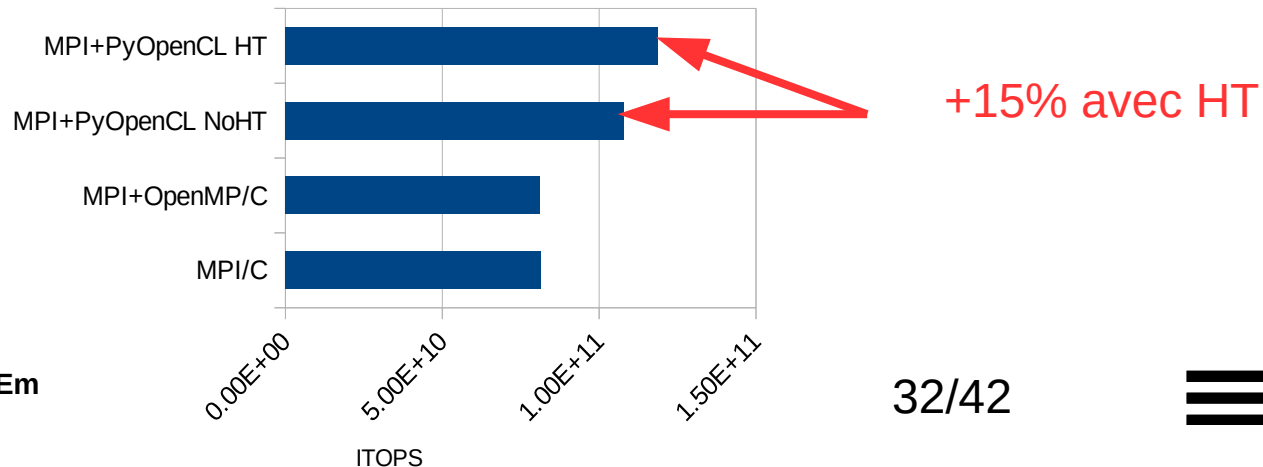
- Contexte des machines :
 - Nœuds équipés de 2 Nethalem x5550 avec l'HyperThreading installé
 - A chaque cœur physique est associé deux cœurs logiques
- Contexte de l'exécution :
 - Chaque nœud exécute 8 tâches MPI :
 - Le noyau distribue normalement efficacement les tâches sur mes cœurs
 - La distribution peut mal se passer :
 - 2 tâches distribuées sur 2 cœurs logiques mais associés au même cœur physique
 - Temps partagé efficace mais cœur physique avec une double « charge »
 - L'exécution prend le double du temps pour ce job, donc « plombe » les autres

Quel « miracle » entre jours 1 et 2 ?

« Localisation » des processus

- Localisation : forçage de l'exécution sur une ressource
 - Exemple : exécution sur le cœur 0 du programme `MyTest`
 - `hwloc-bind -p pu:0 MyTest`
 - Dans le cas présent, code MPI et 7 premiers cœurs :
 - `/usr/bin/time mpirun.openmpi -np 384 -hostfile $MyHostFile hwloc-bind -p pu:0-7 Pi_MPI_FP32_MWC 1000000000000000`
- Une solution : désactiver l'HyperThreading dans le BIOS

– Mais :



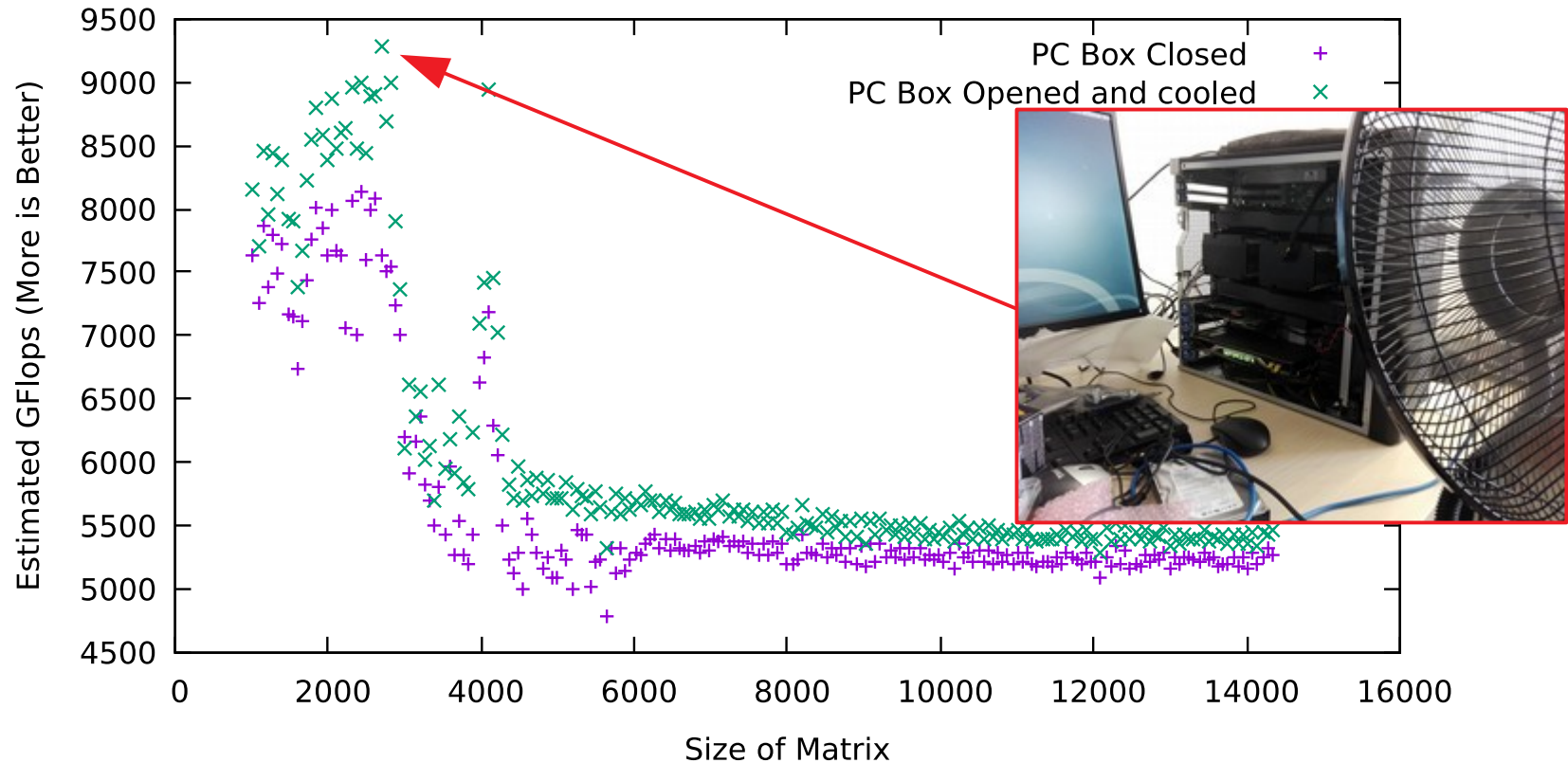
Manque de reproductibilité : L'influence climatique

- Objectif :
 - Évaluation de la performance de xGEMM sur les GPU
- Plate-forme d'expérimentation : matériel & système
 - Une station Precision 7910 bi-sockets, quadri-cœurs
 - 2 GPUs : une Nvidia Quadro K420 & Nvidia GTX 1080 Ti
 - **Un système SIDUS basé sur stretch en amd64**
 - Un programme « maison » exploitant xGEMM en cuBLAS
 - Exploration de différentes tailles de 1024 à plus de 14000

xGEMM sur GPU :

les tests sur GTX 1080 Ti

xGEMM for a Nvidia GTX 1080Ti: performances for cuBLAS implementation



- Mêmes station, cartes, système, et 20 % de différence !
 - Conditions climatiques (et leur connaissance) importantes en expérimentation !

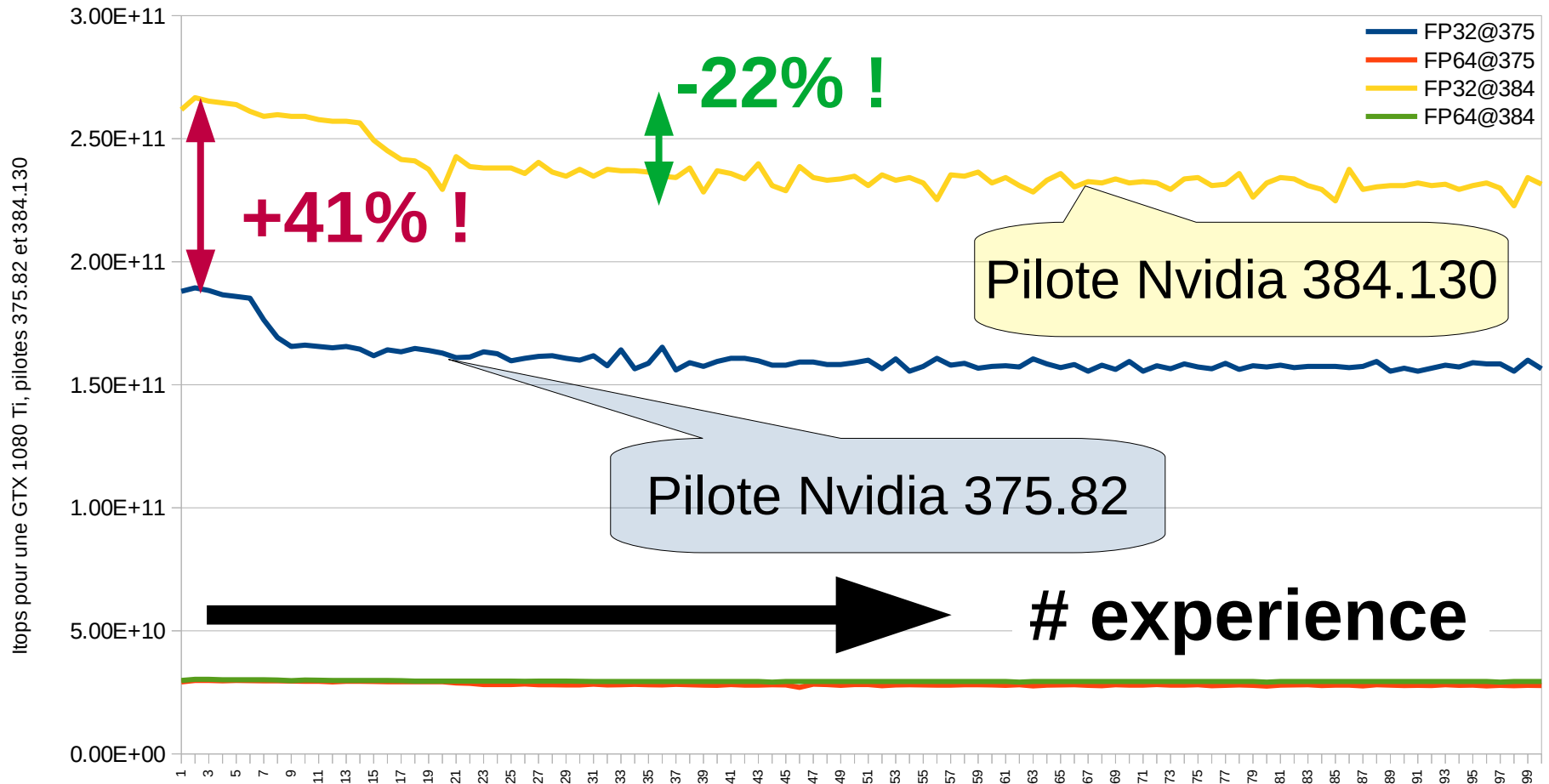
Manque de reproductibilité :

Autres influences

- Objectif :
 - Évaluation de la performance sur GPU pour les mêmes tâches
- Plate-forme d'expérimentation : matériel & système
 - Une station Precision 7910 bi-sockets, quadri-cœurs
 - 2 GPUs : une Nvidia Quadro K420 & **Nvidia GTX 1080 Ti**
 - **Un système SIDUS basé sur stretch en amd64**
 - Deux versions de pilotes Nvidia : 375.82 et 384.130
 - Le programme fétiche Pi Dart Dash en OpenCL
 - 100 exécutions consécutives

Versions de pilotes & expériences

Facteurs de variabilité...



A bien prendre en compte dans l'expérience !

Pourquoi ces variabilités ?

Entre Pilotes, entre exécutions

- Entre versions de pilotes :
 - Nvidia a « bien » travaillé (probablement de la vectorisation à la volée)
 - Attention : gain, uniquement sur ce code test :-(
- Entre exécutions successives :
 - Schéma comparable : pas trop mal, top, décroissance, stabilisation...
 - Explication en 3 lettres : TDP
 - TDP : Thermal Design Power, enveloppe thermique
 - Grosse sollicitation, overclocking sur le GPU
 - Augmentation de la TDP au dessus d'un seuil puis baisse de fréquence...

En SIDUS dans tout ça !

Outil de reproductibilité ?

- Partage instantané d'un système sur un parc de machines
 - Si deux machines n'ont pas le même comportement, c'est sous l'OS
- Manipulation d'un système sans peur du *roll-back*
 - Installation de pilote, logiciel, paramétrage noyau, etc.
 - Redémarrage et un OS tout neuf !
- Conservation d'un système complet en un *snapshot*
 - Commande : `btrfs snapshot /Racine/Ancienne /Racine/Nouvelle`
 - Offre du nouveau NFSroot & pointage des noyaux

Evolutionnements passées & futures de SIDUS

- 2015Q2 : Migration de initramfs-tools à dracut
- 2016Q1 : Addition support UEFI pour machines récentes
- 2016Q3 : Récupération HTTP pour noyau et initrd
- 2017Q2 : Support OverlayFS en plus de AUFS
- 2018Q1 : Exploitation iPXE pour *bypasser* le TFTP local
- 2018Q4 : Support ARM64 pour HP Apollo70 via grub
- 2019Q1 : Démarrage SIDUS over Internet par VPN
- 2020Q3 : Sidus4Labs, le « tout-en-un » avec 3 distros...

Conclure sur SIDUS en particulier et la reproductibilité en général...

- Un « grand pas » vers la déduplication ?
 - Le tester, c'est l'adopter : demander au staff technique du PSMN...
- Un « petit pas » pour la reproductibilité ?
 - La double négation : « ... ne peuvent pas ne pas être identiques »
 - Empreinte minimale pour le stockage et le réseau à l'usage
 - Sur réseau local, chargement avec iPXE et HTTP
 - Sur Internet, chargement avec iPXE et HTTP & OpenVPN
- Pour « bien » reproduire, récupérer (tout) le contexte
- Dans tous les cas : **conservez (ou donnez) votre matériel !**

Démos & Liens utiles

- Démarrage d'un SIDUS au Centre Blaise Pascal
- Démonstration d'une variabilité calculatoire
- Démarrage d'un SIDUS *over Internet*
- Démarrage d'instances Sidus4LABS
 - <https://www.cbp.ens-lyon.fr/doku.php?id=developpement:productions:sidus>
 - 3 instances : Debian Buster 10, Ubuntu 20.04, CentOS 8
 - Des OVA déployables sur n'importe quel VirtualBox
- Le **Cloud@CBP** basé sur SIDUS :
 - <https://www.cbp.ens-lyon.fr/python/forms/CloudCBP>

Appel à dons pour vos matériels obsolètes

- Plus c'est vieux, mieux c'est !
- Même si c'est cassé ça peut servir !
- Et les périphériques (et leur câblerie), c'est important...
- Exploitation pour les ateliers 3IP
 - Introduction Inductive à l'Informatique et au Parallélisme
- Conservation dans la computhèque du CBP
 - Maintien en condition opérationnelle des machines
 - Exemple d'un K6 démarrant sous SIDUS sur Youtube