Centre Blaise Pascal : son centre d'essais

Emmanuel Quémener, Micaël Calvas

Centre Blaise Pascal, ENS-Lyon, Lyon, France

De l'hôtel à projets au centre d'essais

Un hôtel, trois missions

Conférences

- organisation de *Workshops*, séminaires
- nœud CECAM en France
- *Computational Physics* aux Houches

Formations

- internationales : Erasmus Mundus
- continues : toutes disciplines
- initiales : de L3 à M2

Projets

- Hébergement "physique"
- Gestion de projet : forge
- "Paillasses numériques"

Un centre d'essais, trois quêtes

Scalabilité

- traitement : du cluster au GPU
- transport : du série à l'Omnipath
- stockage : du zRam au GlusterFS

Reproductiblité

- dans l'espace : systèmes différents
- dans le temps : répétabilité
- avec définition de "systèmes socle"

Simplicité

- de déploiement d'OS : SIDUS
- d'accès au stockage
- de traitement : HPDA@TheEdge

Quoi ? Petite analogie aéronautique

Compagnie aérienne

exploite

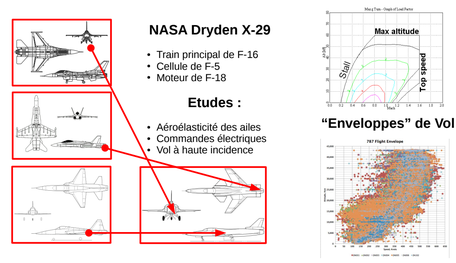
Avionneur

demande

Centre d'essai en vols
Dryden de la Nasa

explore

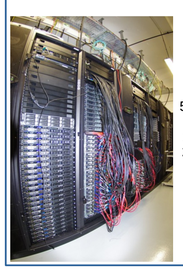
Recycler, Réintégrer
Explorer de nouveaux domaines



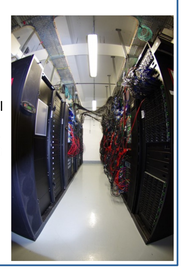
Laboratoire de l'ENS-Lyon & des environs

exploite

Centre de Calcul de l'ENS-Lyon : PSMN
Pôle Scientifique de Modélisation Numérique

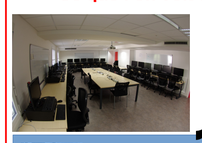


500 nœuds de calcul
pour 8500 cœurs
1.5 PB de stockage
20 baies dans SING



demande

Centre d'essai : Centre Blaise Pascal
Ce que l'on voit & ce que l'on ne voit pas...



- 50 serveurs de machines virtuelles & stockage
- 116 nœuds de calcul pour 736 cœurs de calcul
- 27 commutateurs Ethernet ou Infiniband
- 5 baies dans la nouvelle salle SING
- Une authentification DSI
- Un seul BOFH (+ 1/4) pour s'en occuper !

explore

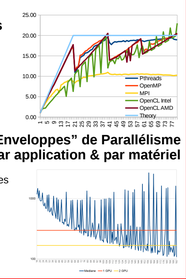
Recycler, Réintégrer
Explorer de nouveaux domaines

Plateaux techniques

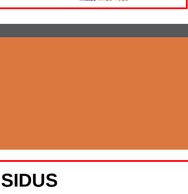
- Vieux clusters du PSMN
- Vieilles baies IN2P3
- Vieux postes de travails
- Prêts de constucteurs
- Achats de GPUs

Études :

- Scalabilité à NP > 500
- Stockage distribué
- Parallélisme des GPU
- Détournements technologiques

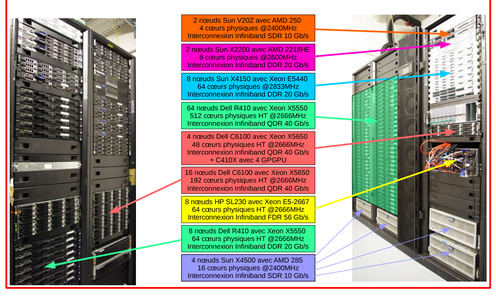


"Enveloppes" de Parallélisme
par application & par matériel

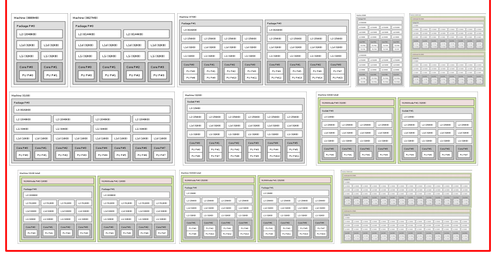


Comment ? Des plateaux techniques & un unique système : SIDUS

Plateau multi-nœuds : 9 grappes
116 nœuds, 4 vitesses réseaux



Plateau multi-cœurs : petit bestiaire
42 types de CPU différents



Plateau myriALUs
Multi-shaders : 72 types de (GP)GPU différents
Accélérateur : 1 Xeon Phi Intel

GPU Gamer : 17

- Nvidia GTX 560 Ti
- Nvidia GTX 680
- Nvidia GTX 690
- Nvidia GTX Titan
- Nvidia GTX 780
- Nvidia GTX 780 Ti
- Nvidia GTX 750
- Nvidia GTX 750 Ti
- Nvidia GTX 960
- Nvidia GTX 970
- Nvidia GTX 980
- Nvidia GTX 980 Ti
- Nvidia GTX 1050 Ti
- Nvidia GTX 1060
- Nvidia GTX 1070
- Nvidia GTX 1080
- Nvidia GTX 1080 Ti
- Nvidia RTX 2080 Ti

GPGPU : 9

- Nvidia Tesla C1060
- Nvidia Tesla M2050
- Nvidia Tesla M2070
- Nvidia Tesla M2090
- Nvidia Tesla K20m
- Nvidia Tesla K40c
- Nvidia Tesla K40m
- Nvidia Tesla K80
- Nvidia Tesla P100

GPU desktop & pro : 27

- NVS 290
- Nvidia FX 4800
- NVS 310
- NVS 315
- Nvidia Quadro 600
- Nvidia Quadro 2000
- Nvidia Quadro 4000
- Nvidia Quadro K2000
- Nvidia Quadro K4000
- Nvidia Quadro K420
- Nvidia Quadro P600
- Nvidia 8400 GS
- Nvidia 8500 GT
- Nvidia 8800 GT
- Nvidia 9500 GT
- Nvidia GT 220
- Nvidia GT 320
- Nvidia GT 430
- Nvidia GT 620
- Nvidia GT 640
- Nvidia GT 710
- Nvidia GT 730
- Nvidia GT 1030
- Nvidia Quadro 2000M
- Nvidia Quadro K4000M
- Nvidia Quadro M2200
- Nvidia MX150

GPU AMD : 18

- HD 4350
- HD 4890
- HD 5850
- HD 5870
- HD 6450
- HD 6670
- Fusion E2-1800 GPU
- HD 7970
- FirePro V5900
- FirePro W5000
- Kaveri A10-7850K GPU
- R7 240
- R9 290
- R9 295X2
- Nano Fury
- R9 Fury
- R9 380
- RX Vega64

Plateau 3IP (prononcez "Trip")
"Introduction Inductive à l'Informatique et au Parallélisme"
Computhèque

Atelier

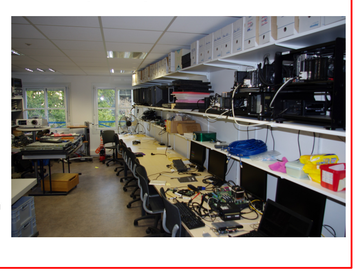
- Diagnostics
- Désassemblage
- Tests unitaires
- (Re)Qualification
- Récupération supports

Refuge

- Machines "ouvertes"
- Machines "exotiques"
- Composants obsolètes

Salle de formation

- Ateliers 3IP
- Fête de la science



Un seul système, SIDUS
"Single Instance Distributing Universal System"
Instance unique distribuant un système universel

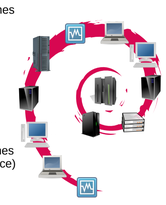
Quoi ?
- Déployer un système simplement sur un parc de machines

Pourquoi ?
- Assurer l'unicité des configurations
- Limiter l'empreinte du système sur les disques

Pour qui ?
- Étudiants, enseignants, chercheurs, ingénieurs, ...

Quand & Où ?
- Centre Blaise Pascal : depuis 2010, près de 200 machines
- PSMN : depuis 2011, plus de 500 nœuds (propre instance)
- Laboratoires : UMPA, LBMC, IGFL, ISA, ...

Comment ?
- Utiliser un partage en réseau d'une arborescence
- Détourner une ruse de LiveCD

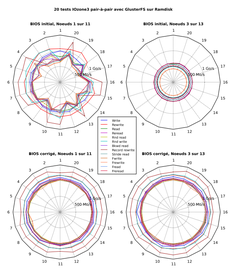


Pour Quoi ? Quelques exemples d'études

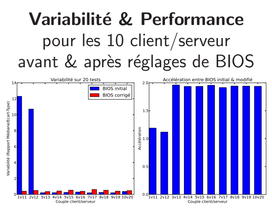
Étude : GlusterFS comme `scratch` en HPC

- **L'objectif** : évaluer les performances de GlusterFS
- **Le contexte** : Equip@Meso, quel scratch ?
- **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
  - 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



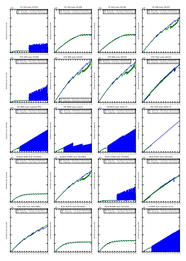
Variabilité & Performance
pour les 10 client/serveur
avant & après réglages de BIOS



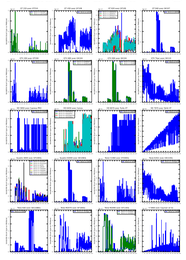
Étude : quel parallélisme massif des GPU ?

- **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
  - **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 :
  - 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



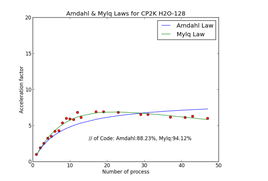
Variabilité de 20 GPU



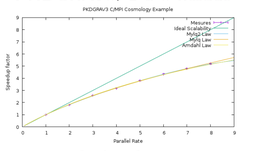
Étude : la loi d'Amdahl représentative ?

- **L'objectif** : évaluer la pertinence de la loi d'Amdahl
- **Le contexte** : codes hybrides OpenMP/MPI
  - 3 applications multi-nœuds, multi-cœurs
  - Deux parallélismes : MPI ou OpenMP/MPI
  - Loi d'Amdahl : $T_n = s + p/n$
  - Loi de Mylq : $T_n = s + p/n + cn + c_2 n^2$
  - s, p, c, c_2 : séquentielle, parallèle, linéaire, quadratique
- **Le banc d'essai :** 3 clusters, 1 OS
  - 48/64 Dell R410 avec 384/512 cœurs physiques
  - 8 Dell C6320 avec 128 cœurs physiques
  - **Socle logiciel SIDUS**
- **Les expériences :** 1 OS, 3 clusters, 3 tests
  - chimie quantique : CP2K, cube 128 H2O
  - astrophysique : PKDGRAV3, 256^3 particules
  - programme "test" : Pi Monte Carlo avec 10^{12} itérations
- **Une interrogation :** quelle scalabilité pour ces codes ?
  - CP2K : plus de gain au delà de 16 nœuds
  - PKDGRAV3 : tassement au delà de 6 nœuds
  - PiMC : baisse au delà de $NP = 400$
- **Les conclusions :**
  - loi d'Amdahl peu représentative des applications
  - loi de Mylq préférable, au moins au 1er ordre
  - scalabilité temporelle mitigée pour codes métier

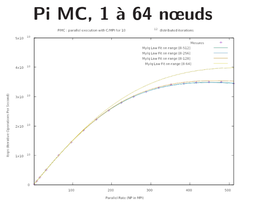
CP2K, 1 à 48 nœuds



PKDGRAV3, 1 à 8 nœuds



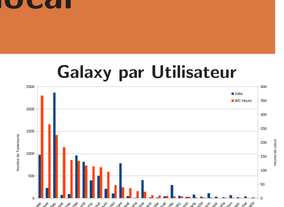
Pi MC, 1 à 64 nœuds



Prototypage : Portail Galaxy local

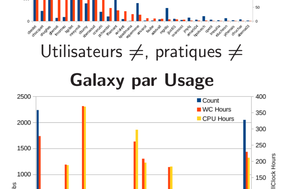
- **L'objectif** : intégration d'un portail Galaxy au centre de calcul
- **Le contexte** :
  - explosion du volume de données des biologistes
  - appropriation du centre de calcul difficile
  - standard émergent de portail de traitement
- **Le banc d'essai :** entre réel & virtuel
  - 1 machine virtuelle pour le portail
  - 1 cluster de 8 nœuds pour les traitements
- **Les étapes :**
  - version 1.0 2015Q2
  - version 2.0 2017Q3 : IB, volume, PostGreSQL
  - version 3.0 2019Q1 : stockage distribué, Shibboleth
- **Les conclusions :**
  - développement en cycle court parfait
  - usages très différents entre laboratoires : pas de panacée...
  - intégration sur infrastructure "centre de calcul" difficile
  - métrologie fine indispensable pour évaluation des besoins

Galaxy par Utilisateur



Utilisateurs ≠, pratiques ≠

Galaxy par Usage

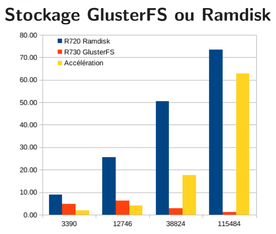


Applications ≠, empreintes système ≠

Étude : Repeat* = "crash applications" ?

- **L'objectif** : quelle infrastructure pour traiter les données ?
- **Le contexte** :
  - workflow trop lourd sur une station de travail
  - portage sur PSMN entraînant une coupure du service
  - investigations poussées sur équipements du CBP
- **Le banc d'essai :** déploiement sur plateformes différents
  - déploiement sur station de travail avec SSD
  - déploiement sur nœud de cluster avec GlusterFS
  - déploiement sur serveur avec Ramdisk
- **Les étapes :**
  - octobre 2013 : station + SSD
  - septembre 2015 : nœud de calcul + GlusterFS
  - octobre 2015 : serveur "musclé" avec un Ramdisk
- **Les conclusions :** une infrastructure nécessaire
  - des traitements générant des millions de petits fichiers
  - un SSD "mort" à la fin de la première campagne
  - un GlusterFS inadapté (comme tout système partagé)
  - une machine à dédier pour le traitement

Exécution d'une même tâche
Stockage GlusterFS ou Ramdisk

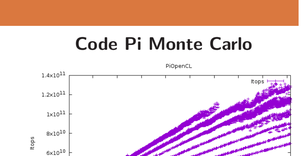


Plus le calcul avance,
plus c'est lent sur le GlusterFS

Caractérisation continue : tous les (GP)GPU

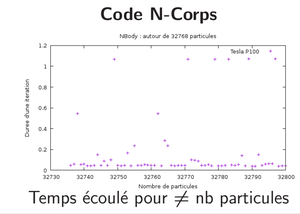
- **L'objectif** : quel "régime de parallélisme" (PR) optimal ?
- **Le contexte** :
  - une nouvelle génération de GPU tous les 2 ans
  - un nombre de cœurs passant de 240 à 4096 en 8 ans
  - une fréquence allant de 500 MHz à 1500 MHz
  - des unités de traitement : *cuda-cores* ou *shader-processors*
- **Le banc d'essai :** 4 codes de tests
  - **xGEMM_*** : le "classique" de BLAS (toutes implémentations)
  - **PiXPU.py** ou **PiOpenCL** : *ALU-bound*, gros grain
  - **NBody.py** : *ALU-bound*, grain fin
  - **Splutter.py** : *MemoryBound*, fonctions atomiques
  - **xTSRV_*** : un assemblage de BLAS (toutes implémentations)
- **Les bizarreries :**
  - optimums de performance : PR = 16x nombre de shaders
  - chutes de performance sur Nvidia : PR = nombre premier
- **Les conclusions :**
  - Nvidia GTX intéressantes mais uniquement pour FP32
  - choix du régime de parallélisme crucial
  - AMD intéressantes mais pénibles à faire fonctionner

Code Pi Monte Carlo



Performances pour ≠ *Parallel Rates*

Code N-Corps



Temps écoulé pour ≠ nb particules

http://www.cbp.ens-lyon.fr/

emmanuel.quemener@ens-lyon.fr micael.calvas@ens-lyon.fr